2

AD-A225 426

NAVAL POSTGRADUATE SCHOOL
Monterey, California

UNITED STATES NAVY · NAVAL POSTGRADUATE SCHOOL ·

THESIS

FUSION OF GROUND-BASED SENSORS FOR OPTIMAL TRACKING OF MILITARY TARGETS

by

John A. Hucks II

December 1989

Thesis Advisor Harold A. Titus

Unclassified

security classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report Approved for public release; distribution is unlimited.
2b Declassification Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)

6a Name of Performing Organization Naval Postgraduate School	6b Office Symbol *(if applicable)* 62	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000		7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding Sponsoring Organization	8b Office Symbol *(if applicable)*	9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*		10 Source of Funding Numbers

Program Element No	Project No	Task No	Work Unit Accession No

11 Title *(include security classification)* FUSION OF GROUND-BASED SENSORS FOR OPTIMAL TRACKING OF MILITARY TARGETS

12 Personal Author(s) John A. Hucks II

13a Type of Report Master's Thesis	13b Time Covered From To	14 Date of Report *(year, month, day)* December 1989	15 Page Count 92

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	Kalman filter, Multiple geometry measurements

19 Abstract *(continue on reverse if necessary and identify by block number)*

Extended Kalman filtering is applied as an extension of the Position Location Reporting System (PLRS) to track a moving target in the XY plane. The application uses four sets of observables which correspond to inputs from a fused-sensor array where the sensors employed are acoustic, seismic, or radar. The nonlinearities to the Kalman filter occur through the measured observables which are; bearings to the target only, ranges to the target only, bearings and ranges to the target, and a Doppler-shifted frequency accompanied by the bearing to that frequency. The observables are nonlinear in their relationships to the Cartesian coordinate states of the filter.

Filter error covariances are portrayed as error ellipsoids about the latest target estimate made by the filter. Rotation of the ellipsoids is accomplished to avoid the cross correlation of the coordinates.The ellipsoids employed are one standard of deviation in the rotated coordinate system and correspond to a constant of probability of target location about the latest Kalman target estimate.

Filtering techniques are evaluated for both stationary and moving observers with arbitrarily moving targets. The objective of creating a user-friendly, personal computer based tracking algorithm is also discussed.

20 Distribution Availability of Abstract ☒ unclassified unlimited ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual Harold A. Titus	22b Telephone *(include Area code)* (408) 646-2560	22c Office Symbol 62Ts

DD FORM 1473,84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete

security classification of this page

Unclassified

Fusion of Ground-Based Sensors for Optimal Tracking of Military Targets

by

John A. Hucks II
Captain, United States Marine Corps
B.S., Lamar University, 1978

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1989

Author: [illegible]
John A. Hucks II

Approved by: [illegible]
Harold A. Titus, Thesis Advisor

[illegible]
Roberto Cristi, Second Reader

[illegible]
John P. Powers, Chairman,
Department of Electrical and Computer Engineering

ABSTRACT

Extended Kalman filtering is applied as an extension of the Position Location Reporting System (PLRS) to track a moving target in the XY plane. The application uses four sets of observables which correspond to inputs from a fused-sensor array where the sensors employed are acoustic, seismic, or radar. The nonlinearities to the Kalman filter occur through the measured observables which are: bearings to the target only, ranges to the target only, bearings and ranges to the target, and a Doppler-shifted frequency accompanied by the bearing to that frequency. The observables are nonlinear in their relationships to the Cartesian coordinate states of the filter.

Filter error covariances are portrayed as error ellipsoids about the latest target estimate made by the filter. Rotation of the ellipsoids is accomplished to avoid the cross correlation of the coordinates.The ellipsoids employed are one standard of deviation in the rotated coordinate system and correspond to a constant of probability of target location about the latest Kalman target estimate.

Filtering techniques are evaluated for both stationary and moving observers with arbitrarily moving targets. The objective of creating a user-friendly, personal computer based tracking algorithm is also discussed.

THESIS DISCLAIMER

The reader is cautioned that the computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made within the time available to ensure that the programs are free of computational or logical errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

The author is deeply indebted to the United States Marine Corps for the opportunity to pursue this work. He is further indebted to Dr. Harold A. Titus, without whose counsel and advise, he would have learned far less. The author also wishes to express gratitude to Dr. Mostafa Ghandahari, who showed him the light in optimal control.

The author wishes to express gratitude to his wife for her patience and understanding through yet another rough period. Last, the author wishes to thank a great friend in Captain Stephen L. Spehn USMC for his insight into what turned out to be a fairly complex problem. Gratitude is also expressed for his collaboration on the error ellipsoid problem contained herein.

To his sons, Nicholas and Patrick, the author dedicates this thesis.

I. INTRODUCTION

The military principle of combat maneuver, in its most basic sense, relies upon the knowledge of the ground commander as to his position and orientation within the combat arena. The units involved in a joint maneuver must coordinate their positions and orientations with each other in order to ensure maximum cooperation and the successful prosecution of the military campaign. Recent history dictates that the knowledge of a tactical military unit position has rested in the individual skill of members of the unit with a map and lensatic compass, from which a position could be determined. Orientation of the unit could be controlled through the accurate association of surrounding prominent terrain features to the map. Once location and orientation have been established, they must be transmitted to higher headquarters in order to ensure the proper adjacent unit coordination. This knowledge, along with tactical intelligence information about the position and orientation of local enemy forces allows the ground commander, through his staff, to command and coordinate support operations. These operations would include indirect fire support, close-air support, deep-air support, and logistics and communication support.

The Position Location Reporting System (PLRS) was created by the Hughes Corporation of Los Angeles for the Marine Corps and Army to ensure, or at least improve, command knowledge of the positions and orientations of friendly tactical elements within the combat arena [Ref. 1]. This system, however, does not track enemy forces forward of the Forward Edge of the Battle Area (FEBA). This can be overcome by the use of an extended Kalman filter algorithm designed as the tracking element with inputs from a fused sensor array. The sensors employed would be restricted to four different types of observables: target bearings, target ranges, both target bearings and target ranges, or a Doppler-shifted frequency signal accompanied by the bearing to that signal. The Kalman filter algorithm would take the sensor inputs and through a conversion process, show observed and estimated target tracks accompanied by a predicted track based on the last estimated position of the target. An addition to the Kalman filter algorithm could then allow for the display of ellipsoids of constant probability (error ellipsoids) about the estimated target positions to show an area in which the target is located to a definite probability.

The operation of PLRS will be discussed, at least partially, in the next chapter. The accumulated knowledge of the PLRS observables (position and location of friendly units) and the observables of the algorithms developed in this thesis could contribute greatly to the knowledge that a potential ground tactical commander could gain about his situation within the combat arena and serve to aid in the tactical decision making process. The knowledge of friendly and enemy forces within a closed combat arena is essential to the successful prosecution of military objectives. This axiom is best summed up by the Chinese philosopher general, Sun Tzu [Ref. 2] who said

> If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle ....

One can easily see the necessity for accumulated combat maneuver intelligence. Since PLRS has no ability to track an enemy force which is forward of the FEBA and concentrates solely on friendly units, a potential tracking algorithm is developed in an attempt to augment the PLRS system. The algorithm is as previously described and based on multiple fused sensors providing measured observables to the tracking element for processing. Four types of observables were considered: target bearings only, target ranges only, both target bearings and target ranges, and a Doppler-shifted frequency accompanied by the bearing to that frequency. These observables are then fed to an extended Kalman filter simulation algorithm to produce the estimated and the predicted tracks.

The extended Kalman filter assumes the existence of nonlinearities between the observables and the system states. The simulations conducted restrict movement of the target to the XY plane and the system states are derived from these dimensions. Nonlinearities to the filter algorithms occur in all of the observables considered relative to the derived system states. An error ellipsoid algorithm is then employed in order to further isolate the target and to attempt to confirm its probable location. The algorithms are then combined using a personal computer-based, user-friendly driver routine in order to acquire the necessary inputs. The total simulation program is then analyzed for performance and an assessment of its tracking capabilities provided.

II. POSITION LOCATION REPORTING SYSTEM

The PLRS system was developed for many different reasons -- among them was the ability to protect friendly tactical elements from fire from friendly sources during combat situations and during peacetime training exercises. Accidents occur every year at the Marine Corps Air Ground Combat Center (MCAGCC), Twentynine Palms, California, which are a direct result of a lack of knowledge as to subordinate unit positions by command elements of friendly units. This is caused by untimely updating over command communication nets. The Combined Arms Exercise (CAX) and the Regimental Firing Exercise (FEX) are commonly used to combine actual movement of troops with live fire in order to impress upon commanders the importance of proper coordination and to increase realism in combat training. All of the movement corridors at MCAGCC are considered as live fire areas and, therefore, the positions of all maneuver elements within them is considered essential to both personnel safety and to mission accomplishment.

The system consists of a containerized master unit composed of generalized computer, communications, and command electronics. This unit is designated as the Master Unit (MU). It is responsible for processing all information inputs into the system and for the display of unit positions, coordination measures, and control information visually. Locally-placed user units are man or machine portable and are designated as User Units (UU). The MU sends a radio message to the UU which then computes a time of arrival (TOA) for the message and transmits this information back to the MU. The MU then uses the arrival time information from the UU to compute a range to the UU. The UU also transmits the local barometric pressure to the MU. The MU then uses the range, bearing, and pressure information to compute a three-dimensional position of the UU. Since the MU will not always have line-of-sight radio contact with all of the local user units, the system relies on the user units to determine the distances between themselves and other locally-placed user units by computation of a TOA for the transmissions of the other units and to report this range information to the MU. Reports are made to the MU through a multi-level relay technique using other user units as necessary.

Within PLRS, the location of each unit is established through a process called trilateration. This involves taking the ranges from three other units with previously established positions and using those ranges to calculate the position of the unlocated unit. To accomplish this the MU uses four simplified discrete Kalman filters. These filters are called the Central Logic Oscillator Control (CLOC) filter to process clock offset and drift rate, the Mean Sea Level (MSL) filter to process an offset calibration for the barometric pressure measurements reported to it, the Track Review and Correction Estimation (TRACE) filter to enter and partially update each UU position and velocity estimate, and an altitude filter to process the barometric pressure data in order to establish and aid in tracking the vertical positions of the local user units.

Tracking of the user units is accomplished by the MU using an internal coordinate system. The internal coordinate system serves as the basis for all coordinate transformation processing. This system is a transverse mercator projection with its central meridian at the longitude of system center. The system essentially takes the curved surface of the earth and projects it stereographically onto a flat plane. This is then called the Local Transverse Mercator (LTM) system. The LTM system produces a slight amount of distortion for long range measurements at distances far from system center. [Ref. 3]

To use the LTM system, the TOA measurements must be converted to LTM ranges by PLRS in real time. This conversion is accomplished in four steps by the MU and is illustrated in Figure 1 on page 5. The four steps are listed as follows:

1. The conversion of the TOA measurement to a true range measurement.
2. The conceptual movement of the units to an average altitude of zero.
3. The projection of the earth onto a flat plane.
4. The movement of the units back to their original altitudes.

The use of PLRS greatly improves command and control efforts at the regimental brigade level and above. By provision of improved manuever unit movement updates and an increase in communications speed, the PLRS concept significantly improves fire support coordination and logistical support efforts. The improvements gained decrease personnel safety risks while allowing for realism in peacetime training, and increased unit effectiveness in combat.

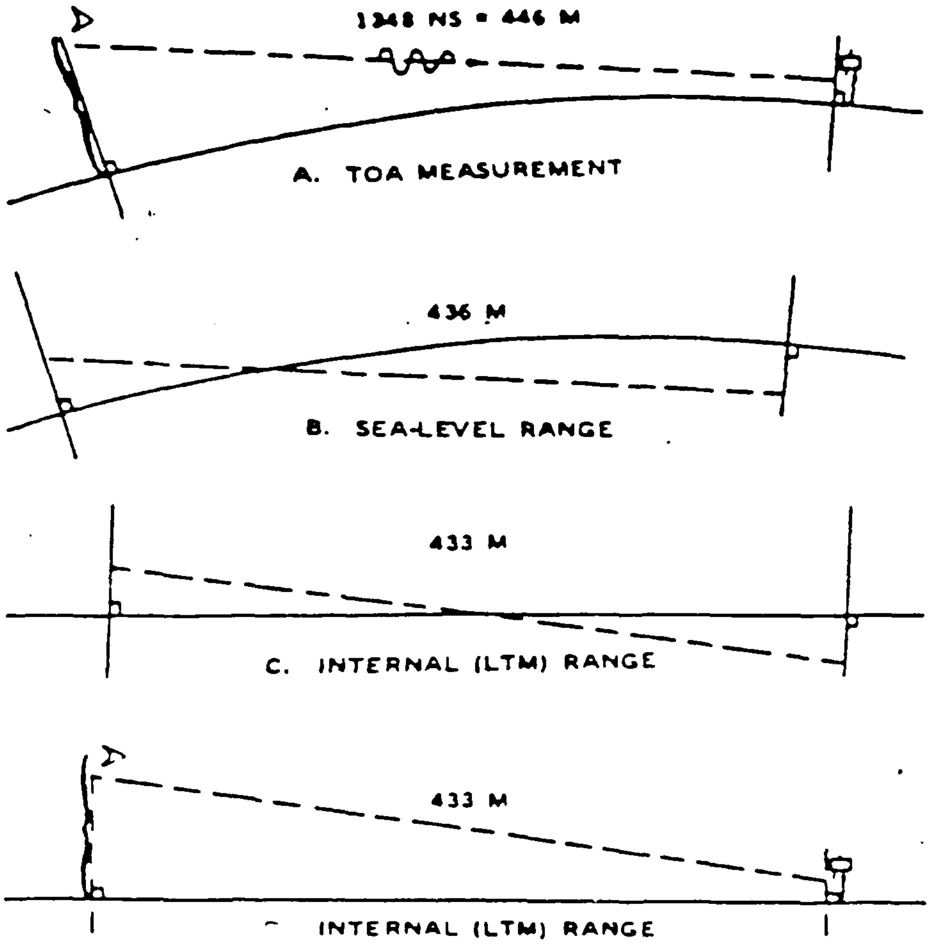

Figure 1. MU Conversion of TOA Data to LTM Range: From [Ref. 3]

III. MULTIPLE GEOMETRY TARGET TRACKING ALGORITHM

In order to study the problem by simulation, three different types of simulation models had to be developed. The sensors, when actually employed in the field, would track a real moving target across the ground. Since a real target does not exist, for the purposes of this simulation, a model had to be developed. The sensor system must also be modeled so that the observables developed from it can be fed to the extended Kalman filter algorithm for processing. The extended Kalman filter must be modeled in a computer algorithm so as to demonstrate its tracking capability. The simulations must then be iterated in order to demonstrate target movement and subsequent updating of the estimated positions and velocities of the modeled target.

A. TARGET MODEL

The motion of the target is restricted to the XY plane. The simulation model uses the basic equation of motion from kinematics

$$s = s_0 + \dot{s}_0 T + \frac{1}{2} a_s T^2 \tag{3.1}$$

This relation is applied in both the X and Y directions and over the total observation time

$$T_{total} = k_{\max} \times T \tag{3.2}$$

where k_{max} is the number of target track positions desired. Thus, a complete set of target positions over a defined time period can be defined. These positions can then serve as the basis for the sensor and Kalman tracker algorithms.

B. SYSTEM MODEL

The system modeled is that of a ground target moving in the XY plane. Certain restrictions were placed on the problem in order to simplify the mathematics. These restrictions were:

- the curvature of the earth is neglected,
- target and sensor movement are restricted to a flat plane, and
- target course and speed inputs are constant (i.e., step inputs).

The observed matrix developed from the sensor simulation algorithm contains no noise, so that Gaussian noise can be added to the observations before entry into the extended Kalman filter algorithm. This supports realism in the simulation in that the observation measurements would contain Gaussian noise.

This is a linear, time-distance system that can be described with equations of motion for constant acceleration in two dimensions. The state space equation is

$$\underline{x}_{k+1} = \underline{\phi}_k \underline{x}_k + \Delta_k \underline{a}_k \tag{3.3}$$

where

$\underline{x}_k$ = state vector (estimation parameter)

$\underline{\phi}_k$ = state transition matrix (describes how the dynamic system states are related)

Δ_k = system noise coefficient matrix

$\underline{a}_k$ = random forcing function.

When the above conditions and Equation (3.3) are incorporated, and the observable system is target bearings only, target ranges only, or both target bearings and target ranges, the state vector is

$$\underline{x}_k = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}. \tag{3.4}$$

When the set of observables is the Doppler-shifted frequency measurement and the associated bearing to that frequency, then the state vector is

$$\underline{x}_k = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ f_0 \end{bmatrix} \tag{3.5}$$

where f_0 is the target transmission rest frequency.

When the following model equations are used and the aforementioned problem conditions incorporated into them, the system state equation can be expanded in matrix form. The model equations are

$$x_{k+1} = x_k + \dot{x}_k T + \frac{1}{2} a_{xk} T^2 \tag{3.6}$$

$$\dot{x}_{k+1} = \dot{x}_k + a_{xk} T^2 \tag{3.7}$$

$$y_{k+1} = y_k + \dot{y}_k T + \frac{1}{2} a_{yk} T^2 \tag{3.8}$$

$$\dot{y}_{k+1} = \dot{y}_k + a_{yk} T^2 \tag{3.9}$$

$$f_{0_{k-1}} = f_{0_k} \tag{3.10}$$

Thus for the systems where the state vector is described by Equation (3.4), the system state equation can be expressed as

$$\begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix}_k + \begin{bmatrix} T^2/2 & 0 \\ T & 0 \\ 0 & T^2/2 \\ 0 & T \end{bmatrix} \begin{bmatrix} a_x \\ a_y \end{bmatrix}_k . \tag{3.11}$$

For the systems where the state vector is defined by Equation (3.5), the system state equation can be expressed as

$$\begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ f_0 \end{bmatrix}_{k+1} = \begin{bmatrix} 1 & T & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & T & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ f_0 \end{bmatrix}_k + \begin{bmatrix} T & T^2/2 & 0 & 0 & 0 \\ 0 & T & 0 & 0 & 0 \\ 0 & 0 & T & T^2/2 & 0 \\ 0 & 0 & 0 & T & 0 \\ 0 & 0 & 0 & 0 & T \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix}_k \tag{3.12}$$

where f_1 through f_5 are the random forcing functions included to account for random changes in speed, direction, or transmission frequency which can occur for a moving target. Each of these random forcing functions must be considered independently in order to ascertain their effect. The forcing functions can be expressed by

$$f_1 = f_1(\gamma_{\theta_t}, \gamma_{v_t}, k) \tag{3.13}$$

$$f_2 = f_2(\gamma_{\theta_t}, \gamma_{v_t}, k) \tag{3.14}$$

$$f_3 = f_3(\gamma_{\theta_t}, \gamma_{v_t}, k) \tag{3.15}$$

$$f_4 = f_4(\gamma_{\theta_t}, \gamma_{v_t}, k) \tag{3.16}$$

$$f_5 = f_5(\gamma_{\dot{f}_0}) \tag{3.17}$$

where $\gamma_{\dot{\theta}_t}$ (for heading), $\gamma_{\dot{v}_t}$ (for speed), and $\gamma_{\dot{f}_0}$ (for the transmission rest frequency) are considered as random changes to the target. These changes are assumed to be independent, zero mean, and piecewise constant rates of change. They have variances defined as

$$\sigma_{v_t}^2 = E[(\gamma_{v_t})^2] \tag{3.18}$$

$$\sigma_{\theta_t}^2 = E[(\gamma_{\theta_t})^2] \tag{3.19}$$

$$\sigma_{\dot{f}_0}^2 = E[(\gamma_{\dot{f}_0})^2]. \tag{3.20}$$

The system noise process for the target tracking and prediction problem is a function of the system noise coefficient matrix Δ_k, and the random forcing function a_k as defined for each of the state vectors. For the state vector of Equation (3.4) this is simply the target acceleration vector. For the state vector of Equation (3.5) a_k is defined by

$$a_k = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \end{bmatrix}. \tag{3.21}$$

The state space representation for the Doppler-shifted frequency observable was developed by Mitschang [Ref. 4].

C. MEASUREMENT MODEL

The problem, as studied, considered four types of observables: bearings to the target only, ranges to the target only, both bearings and ranges to the target, and the Doppler-shifted frequency accompanied by a bearing measurement to that frequency. With recall of the state vectors as defined by Equations (3.4) and (3.5), the measurement model for this problem can be developed. The measurement equation is

$$\underline{z}_k = \underline{H}_k \underline{x}_k + \underline{v}_k \tag{3.22}$$

where

$\underline{z}_k$ = the set of measurements

$\underline{H}_k$ = the observation matrix (noiseless)

$\underline{x}_k$ = the state vector

$\underline{v}_k$ = associated measurement noise.

1. Bearings Only Problem

In this tracking problem, the measurements are lines of bearing as received by sensors located on separate prominent terrain features. The geometry of the problem is shown in Figure 2 on page 11. The problem geometry shows that the relationship of the measurements to the system states is nonlinear. For this case the measurement equation becomes

$$\theta_{nk} = \tan^{-1}\left[\frac{(x_{tk} - x_{nk})}{(y_{tk} - y_{nk})}\right] + v_k \tag{3.23}$$

where

θ_{nk} = lines of bearing from sensor n at time k

x_{tk}, y_{tk} = the X,Y coordinates of the target at time k

x_{nk}, y_{nk} = the X,Y coordinates of sensor n at time k

v_k = the bearing measurement noise.

2. Ranges Only Problem

In this tracking problem, the measurements are straight-line distances to the target gained by ranging sensors located on separate prominent terrain features. The problem geometry is shown in Figure 3 on page 12. This geometry, as in the last case, shows a nonlinear relationship between the measurements and the system states. The measurement equation is

$$r_{nk} = \sqrt{x_{tk} - x_{nk}{}^2 + y_{tk} - y_{nk}{}^2} + v_k \tag{3.24}$$

where r_{nk} = range distances from sensor n at time k.

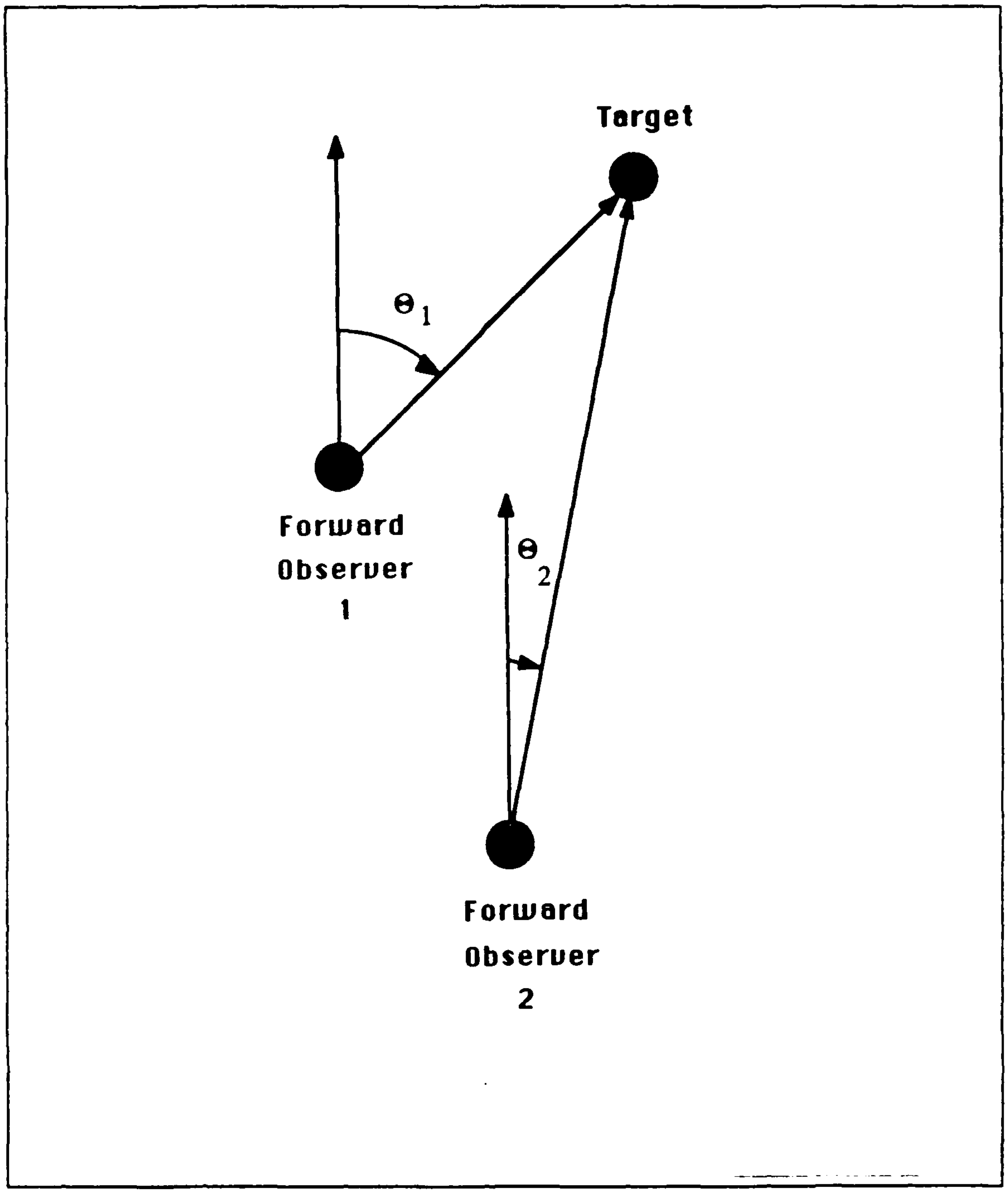

Figure 2. **Bearings Only Geometry:** Measurement Inputs to the Kalman Filter are Restricted to Target Bearings.

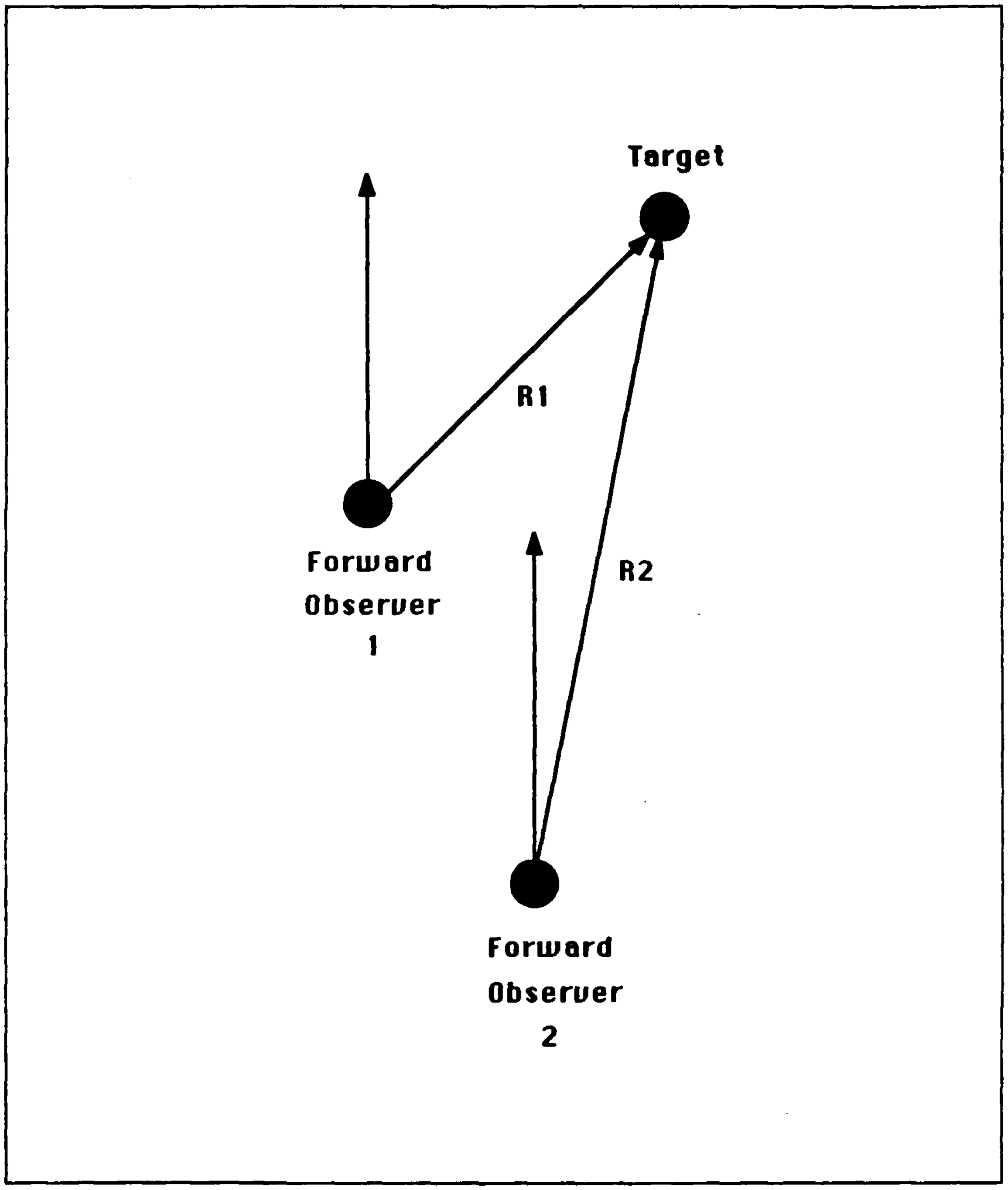

Figure 3. **Ranges Only Geometry:** Measurement Inputs to the Kalman Filter are Restricted to Target Ranges.

3. Bearings and Ranges Problem

The tracking problem now becomes a combination of the first two problems. The measurements are defined as in Equation (3.23) for target bearings and Equation (3.24) for target ranges. The geometry for this problem is shown in Figure 4 on page 14. The observation matrix for this problem is now defined as

$$z_k = \begin{bmatrix} \theta_1 \\ r_1 \\ \theta_2 \\ r_2 \end{bmatrix}_k . \tag{3.25}$$

4. Doppler-Shifted Frequency Problem

In this tracking problem, the observables are a Doppler-shifted frequency accompanied by a bearing measurement to that frequency. The problem geometry as shown in Figure 5 on page 15, along with knowledge of the Doppler equation [Ref. 5], shows that the relationship between the measurements and the system states is again nonlinear. The measurement equation is

$$z_k = \begin{bmatrix} \tan^{-1}\left[\dfrac{x_{tk} - x_{nk}}{y_{tk} - y_{nk}} \right] \\ \dfrac{f_0(k) v_p}{v_p + \left[\dfrac{x_{tk}\dot{x}_{tk} + y_{tk}\dot{y}_{tk}}{\sqrt{x_{tk}^2 + y_{tk}^2}} \right]} \end{bmatrix} + \begin{bmatrix} v_\theta \\ v_f \end{bmatrix}_k \tag{3.26}$$

where

v_p = the velocity of wave propagation (speed of light)

$v_{\theta k}$ = bearing measurement noise

v_{fk} = frequency measurement noise.

This measurement equation is based on a single sensor which is stationary and located at the origin.

The observables in all four measurement cases have been shown to be nonlinear in their relationship to the system states. Processing by the extended Kalman filter is thus required for target tracking and prediction. The operation of the extended Kalman filter will be explained in the next chapter.

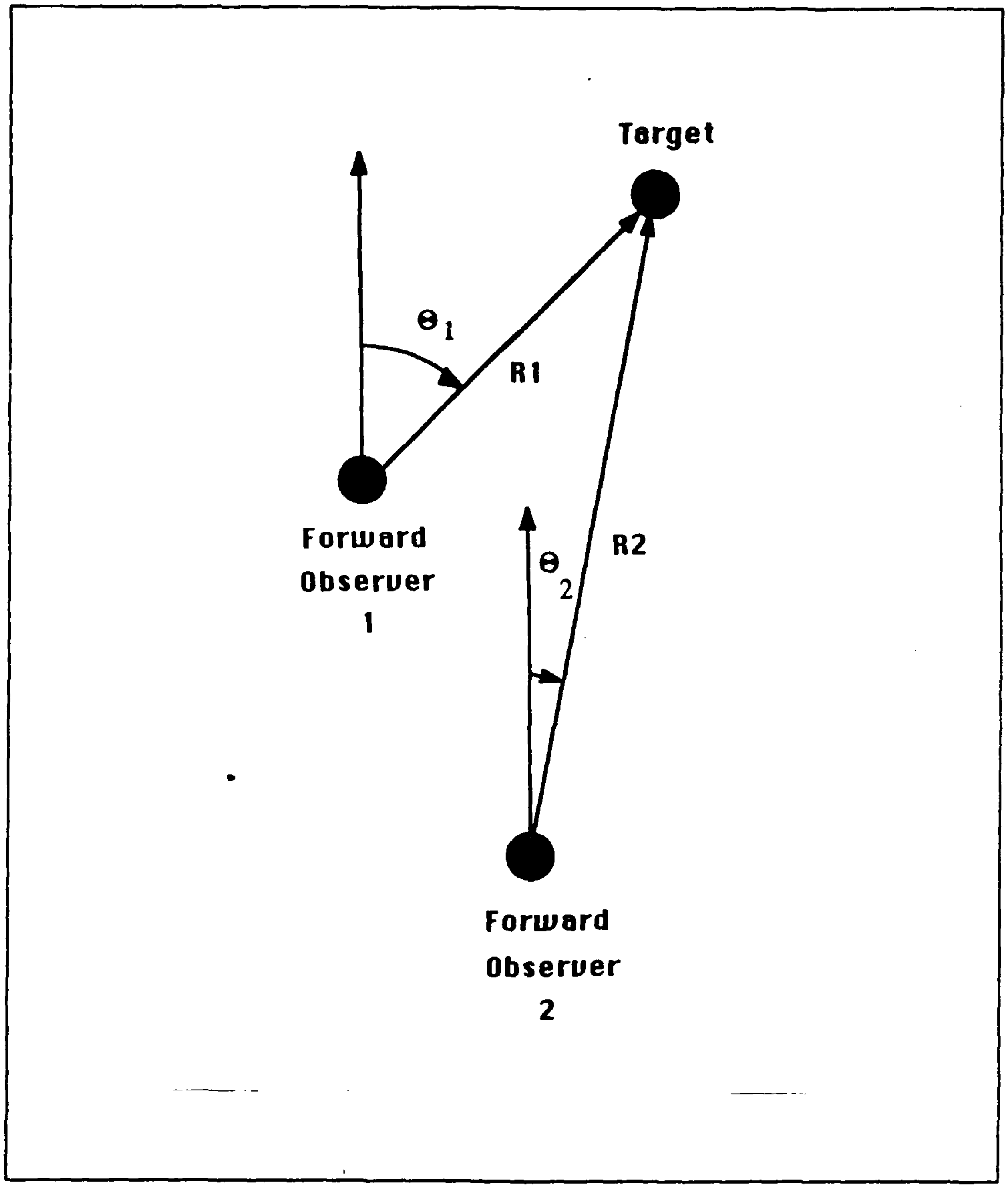

Figure 4. **Bearings and Ranges Geometry:** Measurement Inputs to the Kalman Filter are Restricted to Target Bearings and Target Ranges.

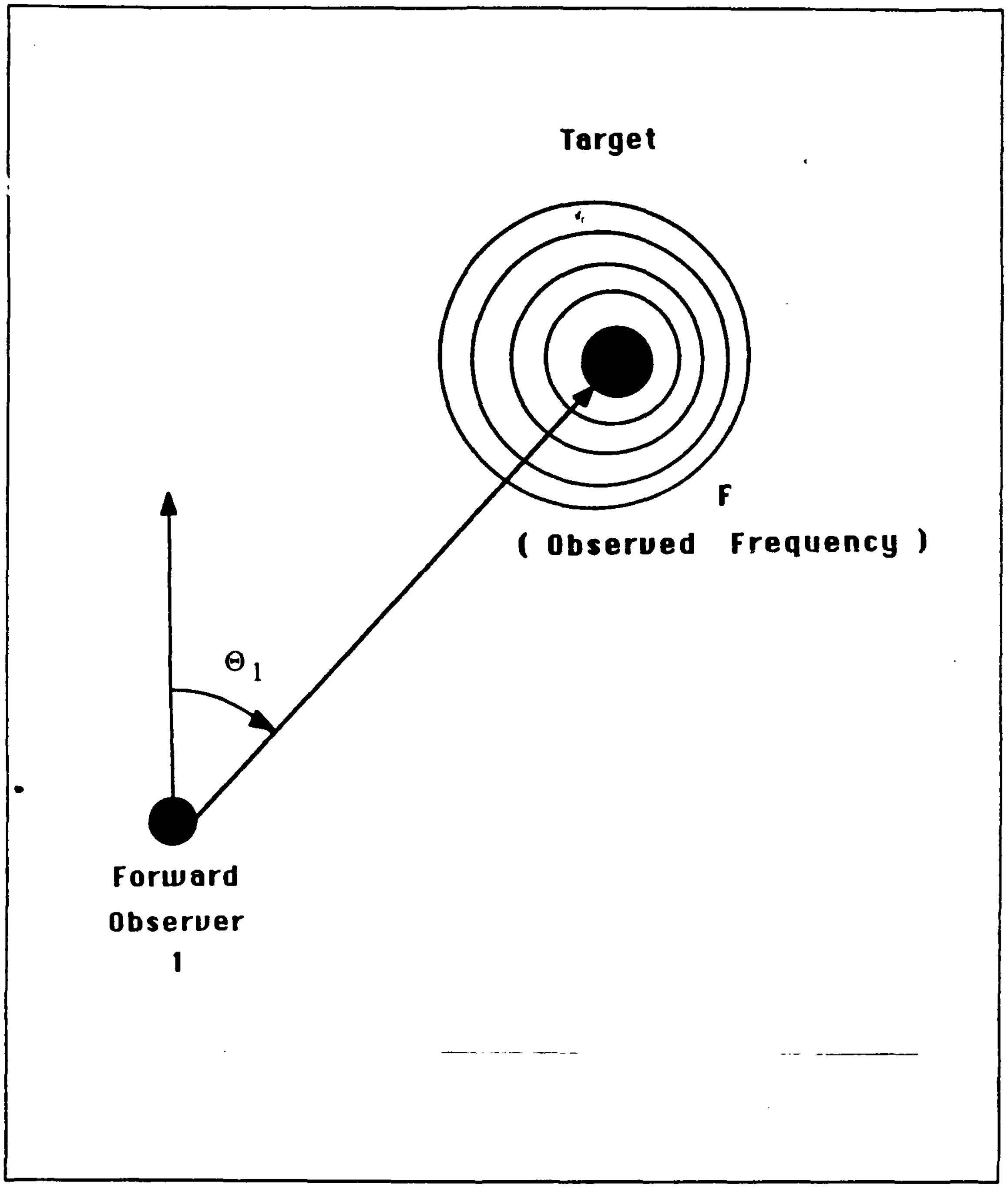

Figure 5. **Doppler Frequency Geometry:** Measurement Inputs to the Kalman Filter are Restricted to a Doppler Shifted Frequency and the Associated Frequency Bearing.

IV. KALMAN FILTER THEORY

The process of estimation of the state vector at the current time, with reference to all previous measurements, is referred to as filtering. An optimal filter optimizes a specific performance measurement which is used to approximate the quality of the estimate. The Kalman filter is a linear optimal filter which minimizes the mean square estimation error between the actual output and the desired output. The filter, in actuality, is a recursive algorithm for the optimal processing of discrete measurements or observations [Ref. 6]. The filter requires an a priori knowledge of the state estimate and its error covariance as well as the current observation. System equations for the Kalman filter are

$$\underline{x}_{k+1} = \underline{\phi}_k \underline{x}_k + \underline{\Delta}_k \underline{a}_k \tag{4.1}$$

and

$$\underline{z}_k = \underline{H}_k \underline{x}_k + \underline{v}_k \tag{4.2}$$

A. THE EXTENDED KALMAN FILTER

From the measurement equations in the last chapter for each of the observable cases, one can see that there is a nonlinear relationship between the measurements used for target tracking and the system state variables. The adaptation of the Kalman filter to a nonlinear application is termed as the extended Kalman filter. A general discussion of the of this type of filter algorithm follows.

Given system and measurement equations of the form

$$\underline{\dot{x}}_k = \underline{f}(\underline{x}_k, k) + \underline{g}(\underline{x}_k, k)\underline{w}_k \tag{4.3}$$

$$\underline{z}_k = \underline{h}(\underline{x}_k, k) + \underline{v}_k \tag{4.4}$$

where

$\underline{f}, \underline{g}, \underline{h}$ are nonlinear functions of the state vector $\underline{x}$,

$\underline{w}_k$ is the plant excitation noise,

$\underline{v}_k$ is the measurement noise.

The plant excitation noise and the measurement noise are assumed as uncorrelated, zero mean, and white. That is

$$\underline{w}_k = N(0, Q), \tag{4.5}$$

$$\underline{v}_k = N(0, R), \tag{4.6}$$

and

$$E[\underline{w}_k, \underline{w}_j^T] = \underline{Q}_k \delta_{kj} \tag{4.7}$$

$$E[\underline{v}_k, \underline{v}_j^T] = \underline{R}_k \delta_{kj} \tag{4.8}$$

where δ_{kj} is the Kronecker delta function such that

$$\delta_{kj} = 1 \qquad (k = j) \tag{4.9}$$

$$\delta_{kj} = 0 \qquad (k \neq j). \tag{4.10}$$

To apply the linear filter Equations (4.1) and (4.2), an expansion of Equations (4.3) and (4.4) is conducted about the best estimate of the state at the current time. This expansion is accomplished with a Taylor series and only the first order terms are kept. With the expansion, Equation (4.3) now yields

$$\underline{x}_{k+1} = \underline{\phi}_k \underline{x}_k + \underline{\Delta}_k \underline{w}_k \tag{4.11}$$

where

$$\underline{\phi}_k = \left. \frac{\partial \underline{f}}{\partial \underline{x}_k} \right|_{\underline{x}_k = \hat{\underline{x}}_k} \tag{4.12}$$

Equation (4.4) now yields

$$\underline{z}_k = \underline{H}_k \underline{x}_k + \underline{v}_k \tag{4.13}$$

where

$$\underline{H}_k = \left. \frac{\partial \underline{h}}{\partial \underline{x}_k} \right|_{\underline{x}_k = \underline{x}'_k} \tag{4.14}$$

The variable $\hat{\underline{x}}_k$ is the estimated value of the state after the k^{th} measurement and the variable $\underline{x}'_k$ is the predicted state value before the k^{th} measurement. These can be specified more completely by

$$\underline{x}'_k = f(\hat{\underline{x}}_{k-1}, k-1). \tag{4.15}$$

The following set of definitions can then be used to derive the linear Kalman filter equations used in the simulation. A state error vector can be defined as

$$\tilde{\underline{x}}_k = \hat{\underline{x}}_k - \underline{x}_k. \tag{4.16}$$

A predicted state error vector can then be defined by

$$\tilde{\underline{x}}'_k = \underline{x}'_k - \underline{x}_k. \tag{4.17}$$

Equations (4.16) and (4.17) allow the definition of the covariance of state error matrix $\underline{P}_k$ by

$$\underline{P}_k = E[\tilde{\underline{x}}_k \tilde{\underline{x}}_k^T] \tag{4.18}$$

and the definition of the predicted covariance of state error matrix $\underline{P}'_k$ by

$$\underline{P}'_k = E[\tilde{\underline{x}}'_k \tilde{\underline{x}}'^T_k]. \tag{4.19}$$

The state excitation matrix $\underline{Q}_k$, as seen in Equation (4.7), is defined by

$$\underline{Q}_k = E[\Delta_k \underline{w}_k \underline{w}_k^T \Delta_k^T]. \tag{4.20}$$

The measurement noise covariance matrix $\underline{R}_k$, as seen in Equation (4.8), is defined by

$$\underline{R}_k = E[\underline{v}_k \underline{v}_k^T]. \tag{4.21}$$

The definitions cited above provide the basis for the construct for the Kalman filter algorithm specified by the set of equations below [Ref. 6]

$$\underline{P}'_{k+1} = \underline{\phi}_k \underline{P}_k \underline{\phi}_k^T + \underline{Q}_k \tag{4.22}$$

$$\underline{G}_k = \underline{P}'_k H_k^T [\underline{H}_k \underline{P}'_k \underline{H}_k^T + \underline{R}_k]^{-1} \tag{4.23}$$

$$\underline{P}_k = [I - \underline{G}_k \underline{H}_k] \underline{P}'_k \tag{4.24}$$

$$\underline{x}'_k = \underline{f}(\hat{\underline{x}}_{k-1}, k) \tag{4.25}$$

$$\underline{z}'_k = \underline{h}(\underline{x}'_k, k) \tag{4.26}$$

$$\hat{\underline{x}}_k = \underline{x}'_k + \underline{G}_k[\underline{z}_k - \underline{z}'_k] \tag{4.27}$$

where

I = the identity matrix

$\underline{G}_k$ = the Kalman gain matrix.

The Q matrix allows for target maneuvering and also accounts for filter model inaccuracies. This greatly decreases the discrepancies which would arise between the system characterization, Equations (4.1) and (4.2), and the true action of the system physically. As the filter algorithm reaches steady-state conditions, the Q matrix also prevents the Kalman gain matrix, $\underline{G}_k$, from approaching zero by ensuring an amount of uncertainty in the predicted covariance of the state error matrix $\underline{P}'_k$.

B. LINEARIZATION OF THE OBSERVATION MATRIX

For purposes of brevity, the H matrix derivations will be done simultaneously for the bearings only, ranges only, and the bearings and ranges observable cases. These are slightly different, however they are based upon the same mathematical premise. Equations (4.4), (4.13), and (4.14) establish the basis for the linearization of the observation matrix H. These equations state that the H matrix is a linearization of the nonlinear function h and is achieved by taking the partial derivative of h with respect to the predicted state.

1. Target-Bearing Measurement Observables Only

The h function using bearing measurements only can be deduced from Equation (3.23) as

$$h(\underline{x}_k, k) = \tan^{-1}\left[\frac{x_{tk} - x_{nk}}{y_{tk} - y_{nk}}\right]. \tag{4.28}$$

Applying the above linearization method,

$$\underline{H}_k = \frac{\partial\left[\tan^{-1}\left[\frac{(x_{tk} - x_{nk})}{(y_{tk} - y_{nk})}\right]\right]}{\partial \underline{x}_k} \tag{4.29}$$

where $\underline{x}_k$ can be recalled for Equation (3.4). Simplifying Equation (4.24) yields

$$\underline{H}_k = \begin{bmatrix} h_{11}\; h_{12}\; h_{13}\; h_{14} \\ h_{21}\; h_{22}\; h_{23}\; h_{24} \end{bmatrix}, \tag{4.30}$$

where

$$h_{n1} = \frac{\partial\left[\tan^{-1}\left[\dfrac{(x_{tk}-x_{nk})}{(y_{tk}-y_{nk})}\right]\right]}{\partial x_{tk}} = \frac{y_{tk}-y_{nk}}{\hat{R}^2_{nk}} \tag{4.31}$$

$$h_{n2} = \frac{\partial\left[\tan^{-1}\left[\dfrac{(x_{tk}-x_{nk})}{(y_{tk}-y_{nk})}\right]\right]}{\partial \dot{x}_{tk}} = 0 \tag{4.32}$$

$$h_{n3} = \frac{\partial\left[\tan^{-1}\left[\dfrac{(x_{tk}-x_{nk})}{(y_{tk}-y_{nk})}\right]\right]}{\partial y_{tk}} = \frac{-(x_{tk}-x_{nk})}{\hat{R}^2_{nk}} \tag{4.33}$$

$$h_{n4} = \frac{\partial\left[\tan^{-1}\left[\dfrac{(x_{tk}-x_{nk})}{(y_{tk}-y_{nk})}\right]\right]}{\partial \dot{y}_{tk}} = 0. \tag{4.34}$$

From Equations (4.31) through (4.34)

$$h_{11} = \frac{y_{tk}-y_{nk}}{\hat{R}^2_{nk}} \tag{4.35}$$

$$h_{12} = 0 \tag{4.36}$$

$$h_{13} = \frac{-(x_{tk}-x_{nk})}{\hat{R}^2_{nk}} \tag{4.37}$$

$$h_{14} = 0 \tag{4.38}$$

where the range estimate squared $\hat{R}^2_{nk}$ is

$$\hat{R}^2_{nk} = (x_{t(k,k-1)} - x_{nk})^2 + (y_{t(k,k-1)} - y_{nk})^2. \tag{4.39}$$

The second row of the matrix is completed in a similar manner.

2. Target-Range Measurement Observables Only

The target-range-measurement-only h function can be deduced from Equation (3.24) as

$$h(\underline{x}_k, k) = \sqrt{(x_{tk} - x_{nk})^2 + (y_{tk} - y_{nk})^2}\,. \tag{4.40}$$

Applying the linearization method again yields

$$\underline{H}_k = \frac{\partial\left[\sqrt{(x_{tk} - x_{nk})^2 + (y_{tk} - y_{nk})^2}\,\right]}{\partial \underline{x}_k} \tag{4.41}$$

where again, $\underline{x}_k$ is recalled from Equation (3.4). Simplification of Equation (4.36) yields

$$\underline{H}_k = \begin{bmatrix} h_{11}\ h_{12}\ h_{13}\ h_{14} \\ h_{21}\ h_{22}\ h_{23}\ h_{24} \end{bmatrix}, \tag{4.42}$$

where

$$h_{n1} = \frac{\partial\left[\sqrt{(x_{tk} - x_{nk})^2 + (y_{tk} - y_{nk})^2}\,\right]}{\partial x_{tk}} = \frac{y_{tk} - y_{nk}}{\hat{R}^2_{nk}} \tag{4.43}$$

$$h_{n2} = \frac{\partial\left[\sqrt{(x_{tk} - x_{nk})^2 + (y_{tk} - y_{nk})^2}\,\right]}{\partial \dot{x}_{tk}} = 0 \tag{4.44}$$

$$h_{n3} = \frac{\partial\left[\sqrt{(x_{tk} - x_{nk})^2 + (y_{tk} - y_{nk})^2}\,\right]}{\partial y_{tk}} = \frac{-(x_{tk} - x_{nk})}{\hat{R}^2_{nk}} \tag{4.45}$$

$$h_{n4} = \frac{\partial\left[\sqrt{(x_{tk} - x_{nk})^2 + (y_{tk} - y_{nk})^2}\,\right]}{\partial \dot{y}_{tk}} = 0. \tag{4.46}$$

Thus from Equations (4.43) through (4.46)

$$h_{11} = \frac{x_{tk} - x_{nk}}{\hat{R}_{nk}^2} \tag{4.47}$$

$$h_{12} = 0 \tag{4.48}$$

$$h_{13} = \frac{(y_{tk} - y_{nk})}{\hat{R}_{nk}^2} \tag{4.49}$$

$$h_{14} = 0 \tag{4.50}$$

and as before, the second row can be computed in a similar manner.

3. Target-Bearing and Range Measurement Observables

The h functions for the case where the observables are both the target bearings and the target ranges are the same as those derived in both of the previous cases. These cases, however, are now combined. The corresponding H matrix is

$$\underline{H}_k = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{bmatrix}, \tag{4.51}$$

where the first and third row of the matrix are the first and second rows of the matrix defined in Equation (4.30) replaced in a one-for-one correspondence. The second and fourth rows of the matrix are the first and second rows of the matrix defined in Equation (4.42) replaced one-for-one.

4. Doppler-Shifted Frequency Observable

The h function for the Doppler-shifted frequency observable and the associated bearing is somewhat more detailed. Using the state vector defined by Equation (3.5) and the measurements obtained from Equation (3.26), one can deduce the following shifted

Doppler h function

$$h(\underline{x}_k, k) = \begin{bmatrix} \theta_k \\ f_k \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left[\dfrac{x_{tk} - x_k}{y_{tk} - y_k} \right] \\ \dfrac{f_0(k) v_p}{v_p + \left[\dfrac{x_{tk}\dot{x}_{tk} + y_{tk}\dot{y}_{tk}}{\sqrt{x_{tk}^2 + y_{tk}^2}} \right]} \end{bmatrix}. \tag{4.52}$$

Application of the linearization process yields

$$\underline{H}_k = \begin{bmatrix} h_{11}\ h_{12}\ h_{13}\ h_{14}\ h_{15} \\ h_{21}\ h_{22}\ h_{23}\ h_{24}\ h_{25} \end{bmatrix}. \tag{4.53}$$

where using the notation of Equation (4.15)

$$h_{11} = \frac{\partial \theta_k}{\partial x_{tk}} = \frac{-y'_{tk}}{x'^2_{tk} + y'^2_{tk}} \tag{4.54}$$

$$h_{12} = \frac{\partial \theta_k}{\partial \dot{x}_{tk}} = 0 \tag{4.55}$$

$$h_{13} = \frac{\partial \theta_k}{\partial y_{tk}} = \frac{x'_{tk}}{x'^2_{tk} + y'^2_{tk}} \tag{4.56}$$

$$h_{14} = \frac{\partial \theta_k}{\partial \dot{y}_{tk}} = 0 \tag{4.57}$$

$$h_{15} = \frac{\partial \theta_k}{\partial f_0} = 0 \tag{4.58}$$

$$h_{21} = \frac{\partial f_k}{\partial x_{tk}} = \frac{-f'^2_k y'_{tk}[y'_{tk}\dot{x}'_{tk} - x'_{tk}\dot{y}'_{tk}]}{f'_{0k} v_p[(x'^2_{tk} + y'^2_{tk})^{3/2}]} \tag{4.59}$$

$$h_{22} = \frac{\partial f_k}{\partial \dot{x}_{tk}} = \frac{-f'^2_k x'_{tk}}{f'_{0k} v_p[(x'^2_{tk} + y'^2_{tk})^{1/2}]} \tag{4.60}$$

$$h_{23} = \frac{\partial f_k}{\partial \dot{y}_{tk}} = \frac{-f'^2_k x'_{tk}[x'_{tk}\dot{y}'_{tk} - y'_{tk}\dot{x}'_{tk}]}{f'_{0k}v_p[(x'^2_{tk} + y'^2_{tk})^{3/2}]} \tag{4.61}$$

$$h_{24} = \frac{\partial f_k}{\partial \dot{y}_{tk}} = \frac{-f'^2_k y'_{tk}}{f'_{0k}v_p[(x'^2_{tk} + y'^2_{tk})^{1/2}]} \tag{4.62}$$

$$h_{25} = \frac{\partial f_k}{\partial f_0} = \frac{f'_k}{f'_{0k}} \tag{4.63}$$

Having linearized the observation matrices for all of the observable cases, the linear Kalman filter equations can be utilized.

C. ACQUIRED NOISE PROCESSES (R AND Q MATRICES)

Calculation of the covariance of state error matrix, $\underline{P}_k$, and the Kalman gain matrix $\underline{G}_k$, requires the specification of covariance matrices for the associated uncorrelated noise processes $\underline{a}_k$ and $\underline{v}_k$ as used in this study. The covariance matrix for the measurement noise process $\underline{v}_k$ is found in Equation (4.8). $\underline{R}_k$ is defined as the state measurement noise covariance matrix. This matrix is based on the accuracy of the sensors employed and accounts for unknown disturbances in the plant model.

The state excitation matrix $\underline{Q}_k$ represents the system noise process and is a function of the system noise coefficient matrix Δ_k and the random forcing function $\underline{a}_k$. . Thus

$$\underline{Q}_k = [\Delta_k \underline{Q}'_k \Delta_k^T] \tag{4.64}$$

where $\underline{Q}'_k$, from Equation (4.20), is defined as

$$\underline{Q}'_k = E[\underline{a}_k \underline{a}_k^T] = \begin{bmatrix} E(a_{xk}^2) & E(a_{yk}a_{xk}) \\ E(a_{xk}a_{yk}) & E(a_{yk}^2) \end{bmatrix}. \tag{4.65}$$

This matrix is valid for the bearings and ranges observables and allows for any random target maneuvers and any inaccuracies in the system model.

The derivation of the Q matrix for the Doppler observable is slightly more complex however, and is based on the random disturbances f_1 through f_5 as defined by Equation (3.21). Recall that γ_{θ_t}, $\gamma_{\dot{v}_t}$, and γ_{f_0} were the random changes to the target and were assumed as independent, zero mean, and piecewise continuous. The variances for these changes can be defined as

$$\sigma_{v_t}^2 = E[\gamma_{v_t}^2] \quad (4.66)$$

$$\sigma_{\theta_t}^2 = E[\gamma_{\theta_t}^2] \quad (4.67)$$

$$\sigma_{f_0}^2 = E[\gamma_{f_0}^2] \quad (4.68)$$

. The standard deviations σ_{v_t}, $\sigma_{\dot{\theta}_t}$, and σ_{f_0} specify typical maneuvering parameters for the target. The Q matrix is found by letting

$$\sigma_x^2 = (\frac{\dot{x}_t}{v_t})^2\sigma_{v_t}^2 + \dot{y}_t^2\sigma_{\theta_t}^2 \quad (4.69)$$

$$\sigma_y^2 = (\frac{\dot{y}_t}{v_t})^2\sigma_{v_t}^2 + \dot{x}_t^2\sigma_{\theta_t}^2 \quad (4.70)$$

$$\sigma_{xy} = \dot{x}_t\dot{y}_t\left[\frac{\sigma_{v_t}}{v_t^2} - \sigma_{\theta_t}^2\right] \quad (4.71)$$

where the states are evaluated at the current state estimates $\hat{\underline{x}}_k$. Substitution of the above expressions into the Q matrix yields

$$\underline{Q}_k = \begin{bmatrix} (T^2/2)^2\sigma_x^2 & (T^3/2)\sigma_x^2 & (T^2/2)\sigma_{xy}^2 & (T^3/2)\sigma_{xy}^2 & 0 \\ q12 & (T^2)\sigma_x^2 & (T^3/2)\sigma_{xy}^2 & (T^2)\sigma_{xy}^2 & 0 \\ q13 & q23 & (T^2/2)^2\sigma_y^2 & (T^3/2)\sigma_y^2 & 0 \\ q14 & q24 & q34 & (T^2)\sigma_y^2 & 0 \\ q15 & q25 & q35 & q45 & T^2\sigma_{f_0}^2 \end{bmatrix} \quad (4.72)$$

This analysis was done by Mitschang and is found in detailed form in Reference 4.

D. PROBLEM PARAMETERS

The following parameters were used in this analysis:

σ_{θ_t} = 6.283 rad/hr

σ_{v_t} = 0.01852 km/hr

$\sigma_{\dot{\theta}_t}^2$ = 0.01096 (rad/min)^2

$\sigma_{\dot{v}_t}^2$ = 0.0001852 (km/min^2)^2.

For the Doppler observable, the parameters were:

$\sigma_{\theta_t} = 0.00017452$ rad/sec

$\sigma_{v_t} = 0.01852$ km/sec

$\sigma^2_{f0} = 0.0005$ Hz/sec.

With the ϕ matrix defined by the system considered as in Chapter III, and the Q, R, and H matrices defined as above, the Kalman filter Equations (4.22) through (4.27) can be employed and the simulations conducted.

V. ERROR ELLIPSOIDS

The position of each unit within the PLRS system is estimated with a certain finite degree of uncertainty about the estimate. The associated uncertainty is expressed in the covariance of error matrix P_k. This represents the sigma squared error deviation about the estimate. The position diagonal terms, P_{11} and P_{33}, represent the variances of the estimate in the X and Y directions respectively. The off-diagonal terms of each represent covariances, the degree of coordinate coupling, and the orientation of the uncertainty in the XY plane.

The errors are normally distributed and there exists a rotated coordinate system in which the orthogonal position components are uncorrelated. This is identical to taking the exponent of the joint probability density function and performing a coordinate transformation to eliminate the cross terms.

The exponent for Gaussian random variables is

$$\frac{x^2}{\sigma^2} - 2r_{xy}\frac{xy}{\sigma_x\sigma_y} + \frac{y^2}{\sigma_y^2}. \qquad (5.1)$$

where r_{xy} is the cross correlation of X and Y. When this expression is set equal to a constant, the geometric interpretation is that of an ellipse which specifies a constant probability of target location about the estimate. The major and minor axis of the ellipse are oriented in the XY coordinate system. In this system a cross correlation between X and Y exists. To eliminate this cross correlation, a coordinate transformation is applied. The positional component X is transformed to a new component X′ and the positional component Y is transformed to a new component Y′ by

$$x' = x\cos\theta + y\sin\theta \qquad (5.2)$$

and

$$x' = y\cos\theta - x\sin\theta \qquad (5.3)$$

where

$$\theta = \frac{1}{2}\tan^{-1}\left[\frac{2\mathrm{cov}(x,y)}{\sigma_x^2 - \sigma_y^2}\right]. \qquad (5.4)$$

This specifies the variances in the new coordinate system (X′Y′) as

$$\sigma_{x'}^2 = \frac{\sigma_x^2 + \sigma_y^2}{2} + \frac{\mathrm{cov}(xy)}{\sin 2\theta} \qquad (5.5)$$

and

$$\sigma_{y'}^2 = \frac{\sigma_x^2 + \sigma_y^2}{2} - \frac{\mathrm{cov}(xy)}{\sin 2\theta}\,. \qquad (5.6)$$

The relation used to plot the resulting ellipsoid, which defines a constant probability of target location about the latest Kalman estimate, results from the transformation. The relation is based on Equation (5.1) and is

$$\frac{x'^2}{\sigma_{x'}^2} + \frac{y'^2}{\sigma_{y'}^2} = K. \qquad (5.7)$$

where K defines the probability of target location.

The error ellipsoid routine employed in this study is based on the above premise and plots ellipsoids respresenting a constant probability of target location about the latest Kalman estimate and uses that estimate as the ellipsoid center. The program was written by Captain Stephen L. Spehn USMC in collaboration with another project. This program plots a single standard of deviation (in the orthogonal coordinates) ellipsoid with a sixty percent probability of target location within the ellipsoid. The major and minor axis of the ellipse may also be plotted in terms of the Mahalanobis distance or the distance in standards of deviation [Ref. 7]. The ellipsoids are increased in size by factors of as much as sixteen for ease of view.

VI. PROGRAM RESULTS

A. PROGRAM LOGIC

The tracking problem solution integrates all four types of geometric concepts of Chapter III. In order to simulate the problem and to attempt a solution, each of the geometric concepts required an individual working subroutine. The subroutines were then integrated using a menu-driven, user-interactive driver file. The driver file was designed to acquire target position and movement information, tracking problem parameters, and to determine the type of problem to be solved.

The program presented herein is an attempt to solve the geometric problem of tracking an arbitrarily moving target in the XY plane. It was written in PC-Matlab in order to use a personal computer-based mathematical algorithm. The system, wher augmented and improved with combinational logic and mapping data, could be used by a maneuver commander in the combat arena as an additional tool in his assessment of the tactical situation. The advantage that could be gained in the knowledge of enemy position and intent would greatly aid in the tactical decision-making process.

1. PC-Matlab Structure

The personal computer-based Matlab program uses a unique program structure that is similar to Fortran. The program, however, uses different terminology in its structure when employing driver routines. The difference in terminology is derived from the way the main Matlab driver programs are written. All Matlab execution files are named with a .m suffix. Thus, all of the file names for this problem contain that suffix. The driver file, marine.m, calls the various subroutines as needed when in operation. The programs written for the study of this problem are included in the Appendix. They are commented throughout in attempt to help the reader to become familiar with their construct with as little effort as possible.

2. Subroutine Structure

The program subroutines each isolate one geometric concept in an attempt to simulate and solve the tracking problem. Each subroutine is written to simulate the target, to simulate the observers (sensors), and to simulate the operation of the extended Kalman filter discussed in Chapter IV. The various steps are commented throughout in an effort to aid the reader in understanding the program and its structure. The names of the variables used in the programs of the Appendix correspond to those of Chapters

III and IV as much as possible. It was felt that this would aid in the understanding of its operation.

B. RESULTS

The results of the program discussed are presented graphically in order to illustrate what the maneuver commander would see if such a system were to be employed on the battlefield. Since the program tracks an arbitrarily moving target with arbitrary observer placement, the target tracks and the observer (sensor) locations were changed with each geometric case. The cases will be presented in the following order: target-bearing observables only employed, target-range observables only employed, both target-bearing and range observables employed, and the Doppler observable with its associated bearing employed.

1. Target-Bearing Observables Only

Results of a tracking problem using only the observed bearings from a fused-sensor array are contained in Figure 6 on page 32 through Figure 9 on page 35. These figures show outputs at various stages of the program. Figure 6 on page 32 shows the actual track of the target, the circles on the diagram, and the noiseless observations, the dots. This graph allows for a comparison of the bearing calculations by the program against the track as chosen in the simulation. This is necessary as a check in the operation of the bearings only subroutine. If the comparison is successful, white noise is added to the bearing measurements before they are submitted to the Kalman algorithm. Figure 7 on page 33 shows the actual and estimated target tracks as well as the error ellipsoids around the Kalman estimates. The simulation iterates and, when in display on the screen, one can see the target move. In this case, the movement is from the upper left to the lower right. The rotation of the error ellipsiods to avoid coordinate cross correlation is clearly evident. The actual positions are again circles, the estimates are crosses, and the error ellipsoids are evident from the plot. Figure 8 on page 34 shows the actual, estimated, and predicted positions of the target. The predictions were obtained by taking the predicted Kalman state estimate and the current Kalman state estimate and performing a subtraction of the velocity values in the X and Y directions and dividing by the observation interval. This produced an estimate of the random forcing function a_k as defined in Chapter III. The estimate was then used in the linear Kalman system equation shown in Equation (4.1) to determine predicted positions based on a constant acceleration. This is consistent with a second-order Kalman analysis in each direction where the acceleration is the random forcing function. The predicted positions are shown as stars in the diagram. Figure 9 on page 35 shows what the potential combat commander would see should the system be employed. The figure is a combination of the actual, estimated, and predicted target tracks as well as the error ellipsoids about the Kalman estimates. The figure also shows the location of the observers, as do the previous figures. These are portrayed as an x in the diagram. Essentially the program was constructed to produce results in response to questions a potential maneuver commander would ask while engaged in combat operations. These questions are:

- Where is the enemy ?
- Where is he going ?

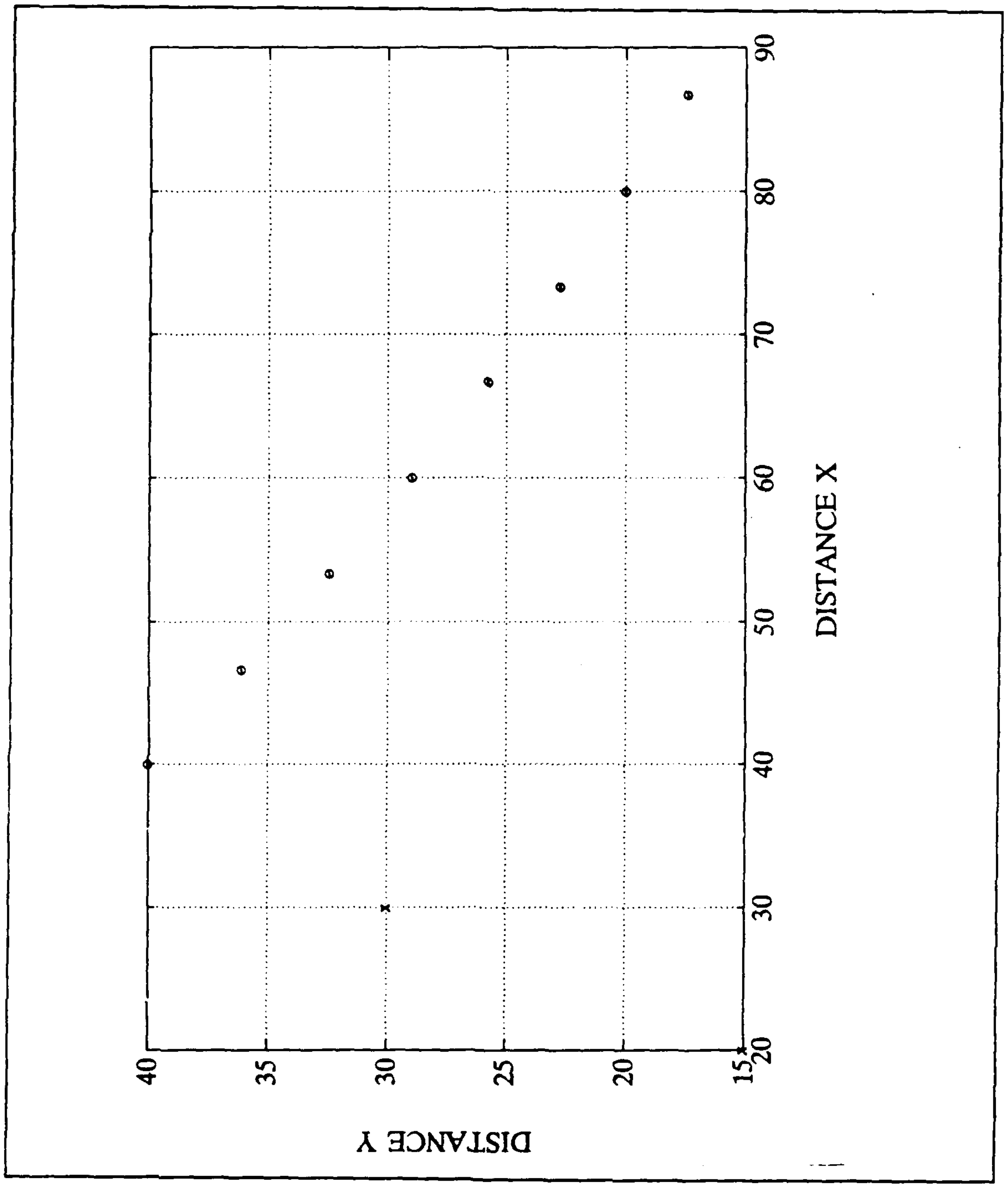

Figure 6. Actual and Observed Tracks: Target-Bearing Observables Only. Actual Track - o, Observed Track - • , Sensor - x.

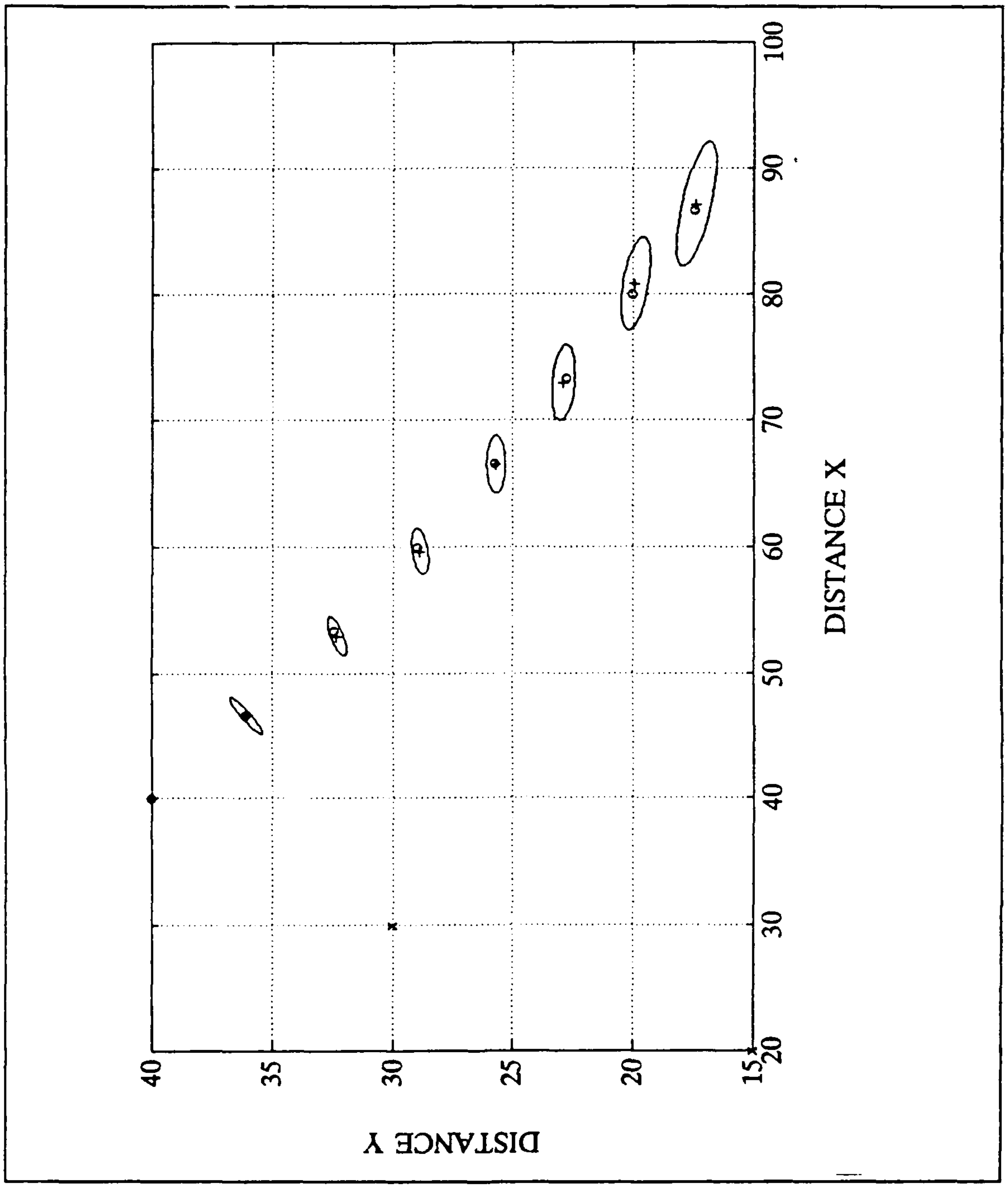

Figure 7. **Actual and Estimated Tracks:** Target-Bearing Observables Only. Actual Track - o, Estimated Track - + . The Plot Also Contains Error Ellipsoids to Aid in Target Location.

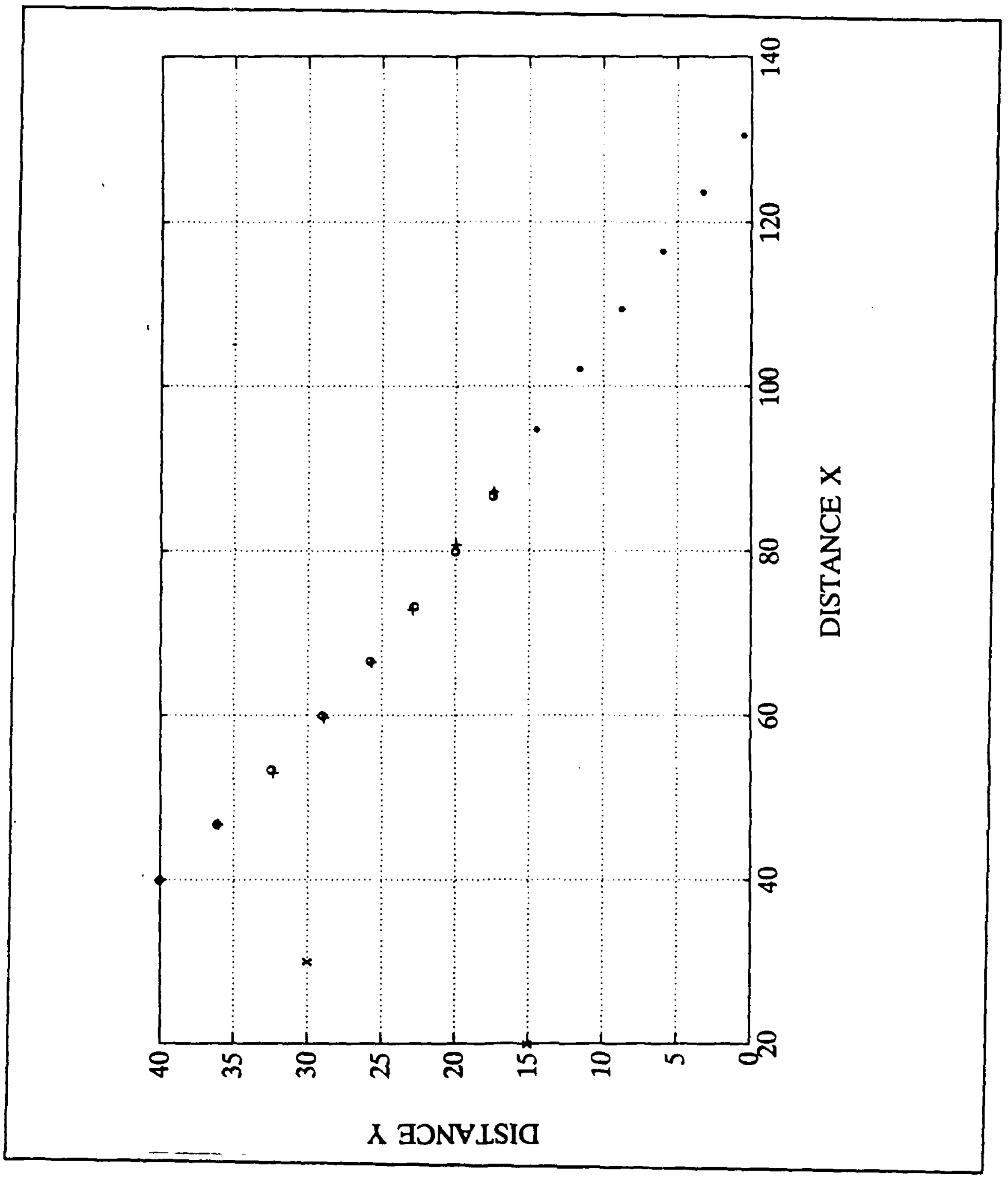

Figure 8. **Actual, Estimated, and Predicted Tracks:** Target-Bearing Observables Only. Actual Track - o, Estimated Track - +, Predicted Track - *.

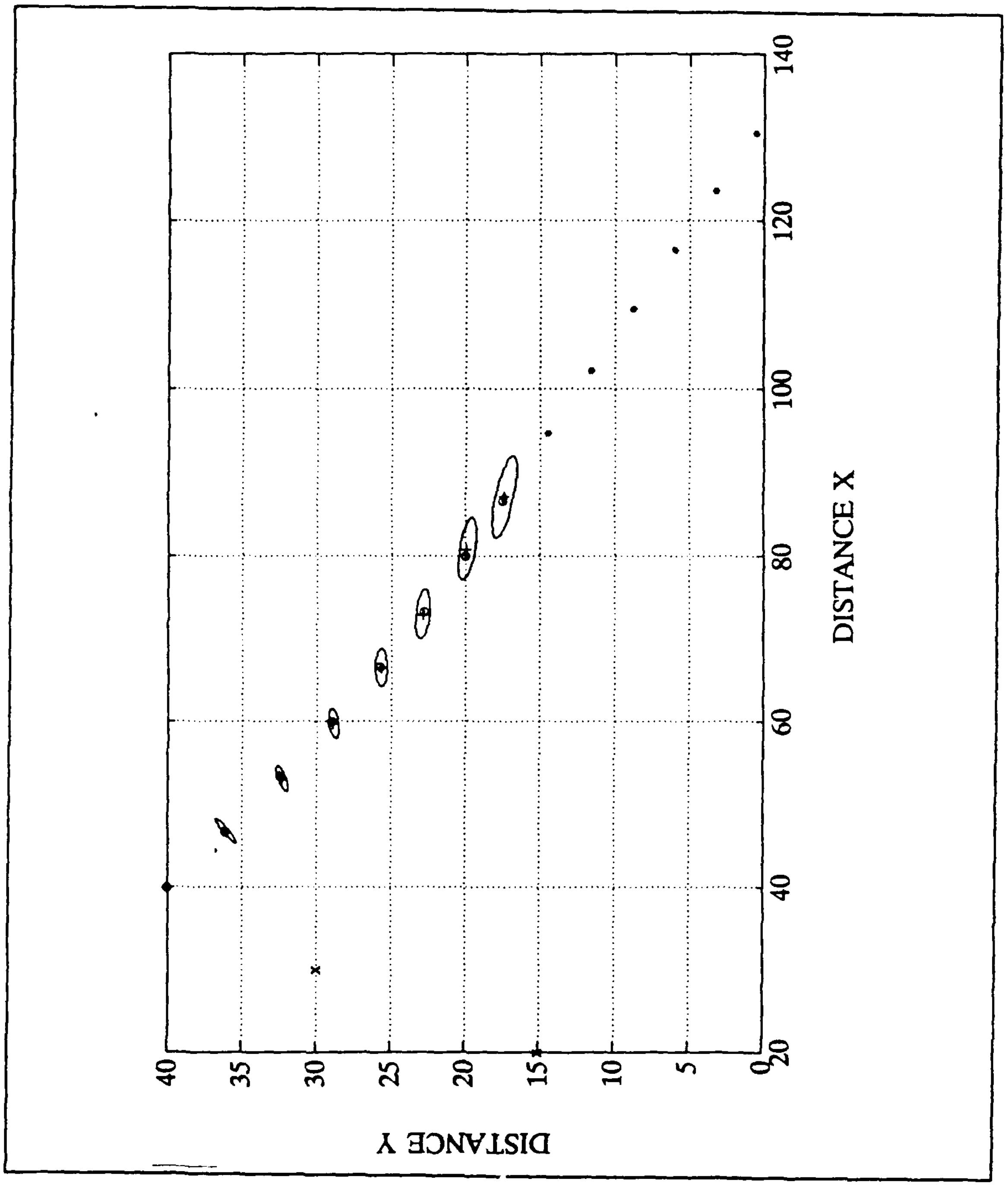

Figure 9. **Total Target Track View:** Target-Bearing Observables Only. Actual Track - o, Observed Track - • , Sensor - x. Predicted Track - *.

2. Target-Range Observables Only

The results of a tracking problem using only target-range observables are shown in Figure 10 on page 37 through Figure 13 on page 40. These figures are in the same order as the ones in the bearings only case. The target track and observer locations have been changed in order to demonstrate the versatility of the program. Figure 10 on page 37 shows the noiseless observations compared to the actual track of the target as chosen in the simulation. Again, if the comparison is successful, Gaussian noise is added before filtering. Figure 11 on page 38 shows the actual and estimated tracks of the target as well as the error ellipsoids about the Kalman estimates. Figure 12 on page 39 shows the actual, estimated and predicted target tracks. When the display is on screen, one can observe the target movement, which in this case would be from left to right across the diagram. Since the observers are nearly colinear with the first estimate, the error ellipsoid shows a substantial bearing error while showing a small range error. This decreases as the target moves away from the observers because the range directions are more accurately determined. Figure 13 on page 40 is a combination of the previous figures which would be used to provide information about the enemy for use in the tactical decision making process.

3. Target-Bearing and Range Observables

The results of a problem employing both target-bearing and range observables are shown in Figure 14 on page 41 through Figure 17 on page 44. The figures are again in the same order as the previous two cases to allow comparison. Figure 14 on page 41 shows the actual target track and the noiseless observations for this case. If the observations favorably compare with the actual positions, white noise is added and the noisy observations filtered by the Kalman algorithm. Figure 15 on page 42 shows the actual and estimated tracks with the error ellipsoids added. The rotation of the ellipsoids to avoid cross correlation of the coordinate variables is particularly evident in this case. Figure 16 on page 43 shows the actual, estimated, and predicted tracks for the target. The target movement in this case was from the lower right to the upper left. Figure 17 on page 44 is a combination of the previous three and would be the result of the program when viewed in the field. Again, it provides answers to relevant command questions.

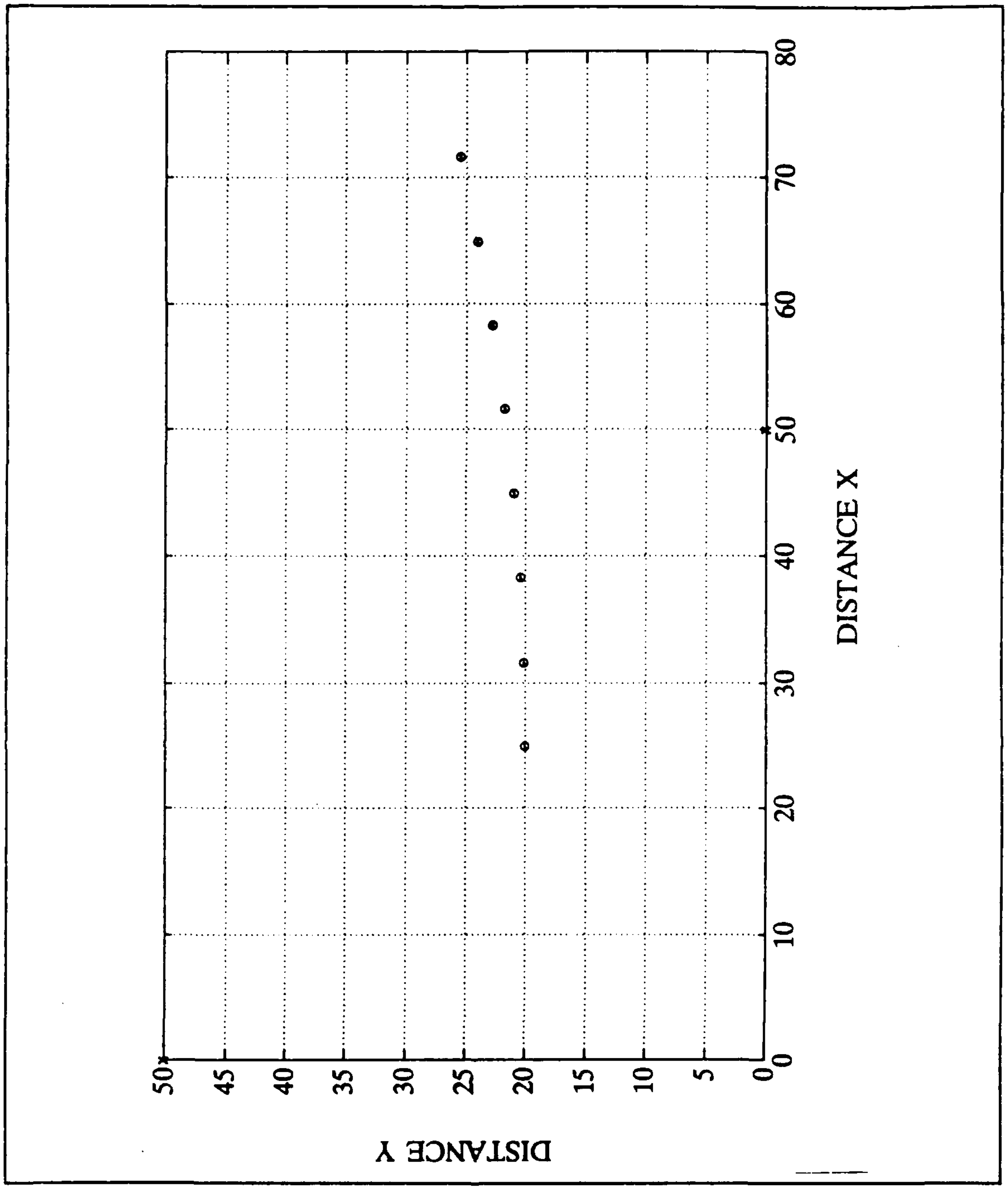

Figure 10. **Actual and Observed Tracks:** Target-Range Observables Only. Actual Track - o, Observed Track - • , Sensor - x.

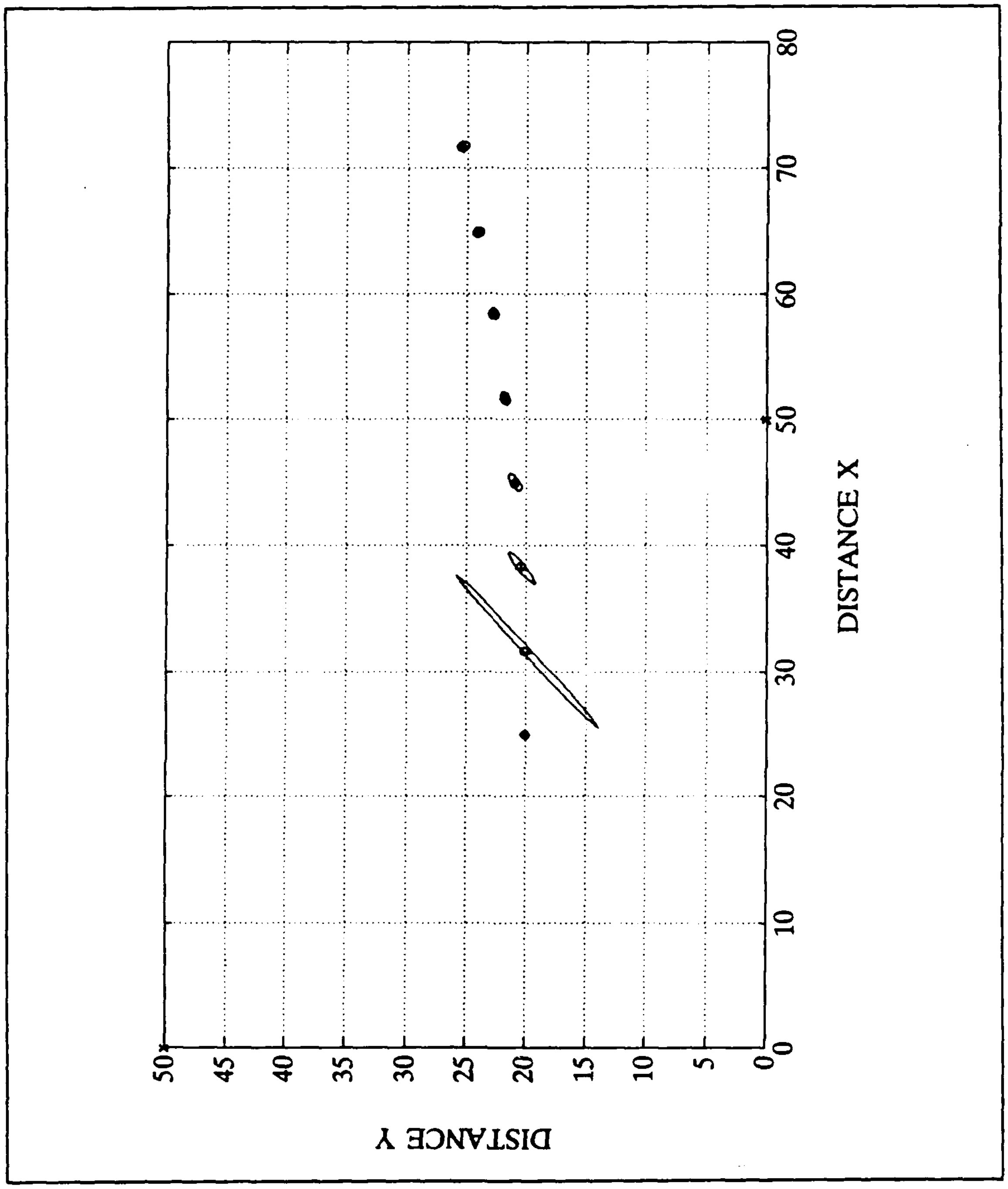

Figure 11. **Actual and Estimated Tracks:** Target-Range Observables Only. Actual Track - o, Estimated Track - + . The Plot Also Contains Error Ellipsoids to Aid in Target Location.

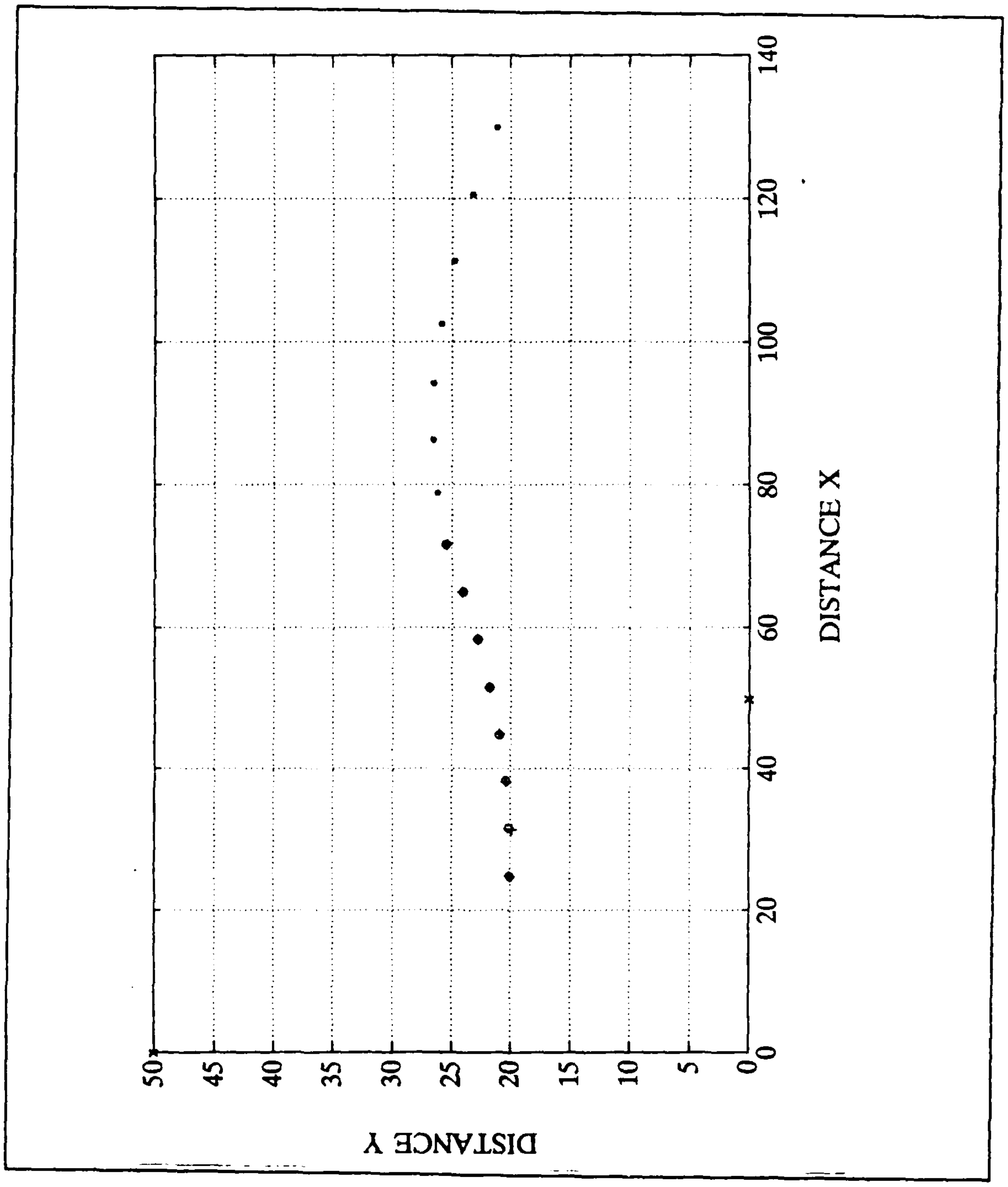

Figure 12. **Actual, Estimated, and Predicted Tracks:** Target-Range Observables Only. Actual Track - o, Estimated Track - +, Predicted Track - *.

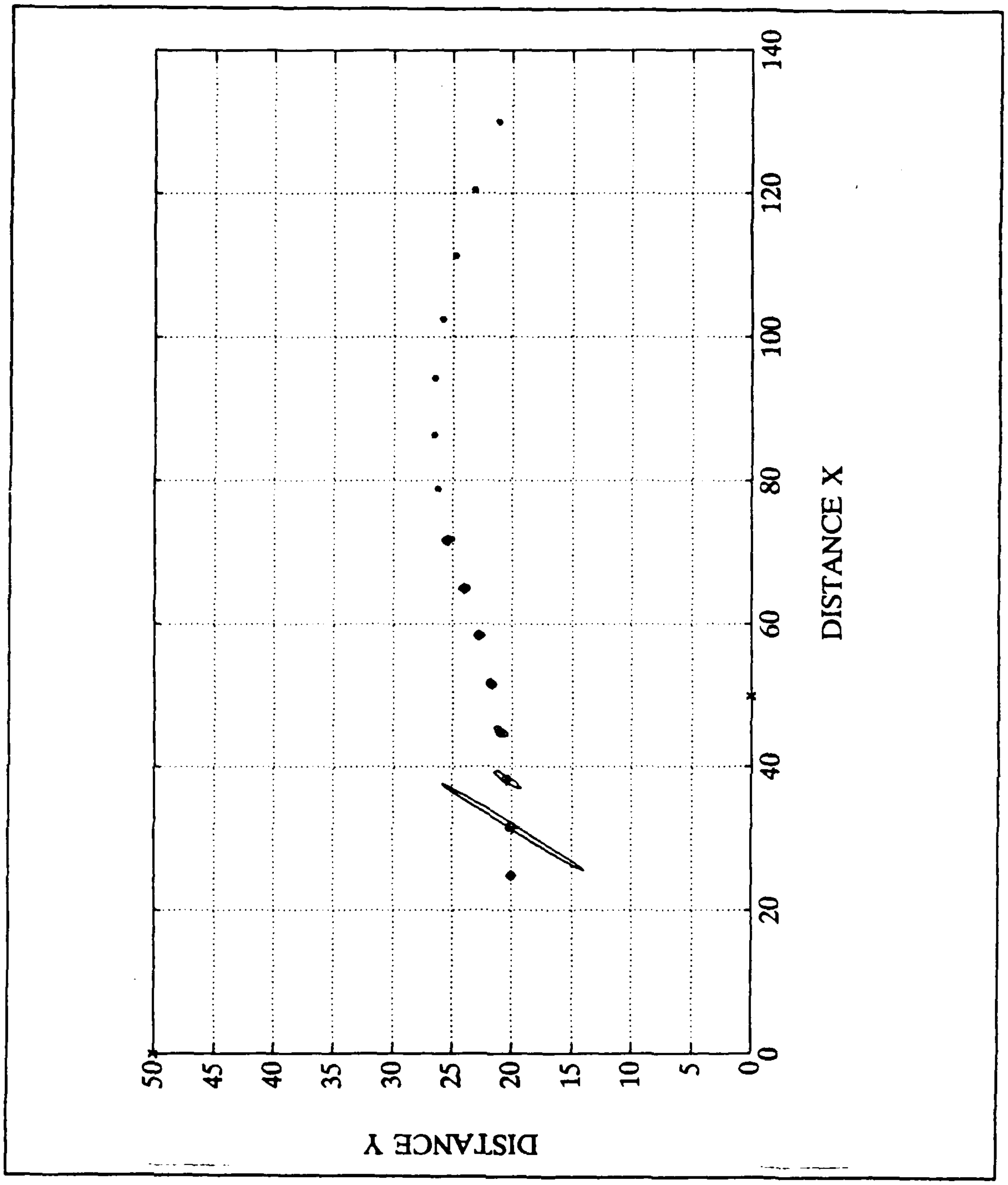

Figure 13. **Total Target Track View:** Target-Range Observables Only. Actual Track - o, Observed Track - • , Sensor - x. Predicted Track - *.

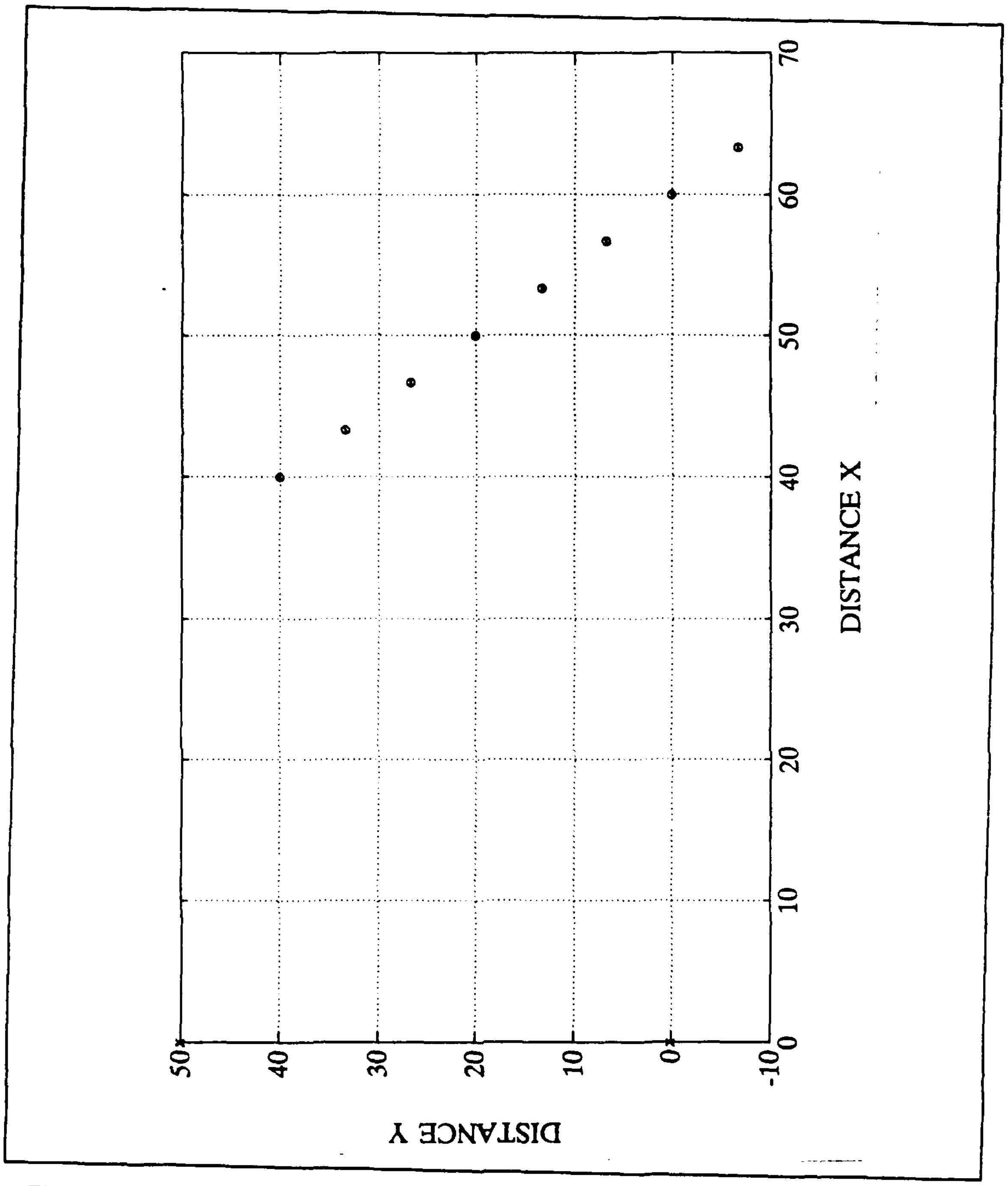

Figure 14. **Actual and Observed Tracks:** Target-Bearing and Range Observables. Actual Track - o, Observed Track - • , Sensor - x.

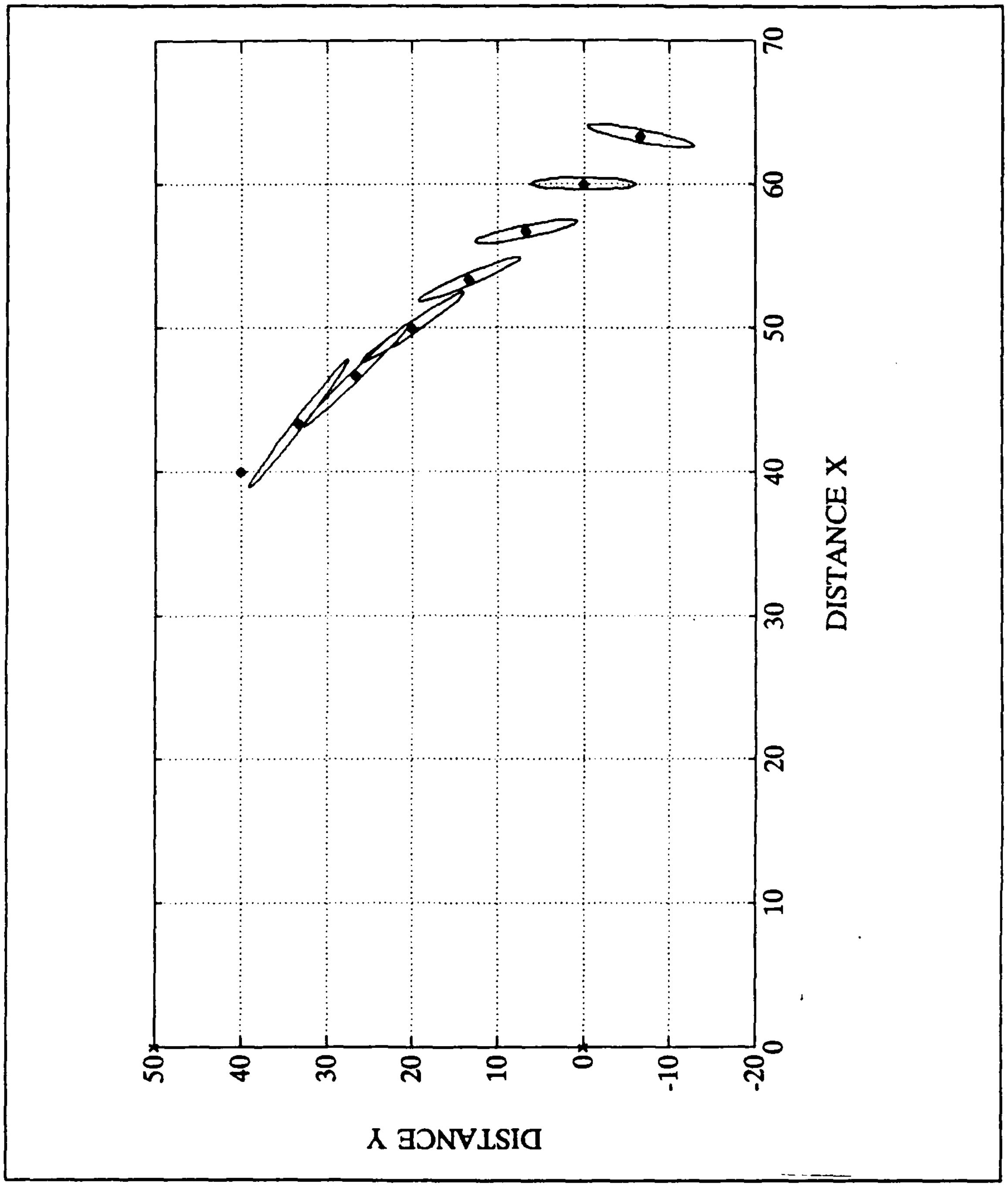

Figure 15. **Actual and Estimated Tracks:** Target-Bearing and Range Observables. Actual Track - o, Estimated Track - + . The Plot Also Contains Error Ellipsoids to Aid in Target Location.

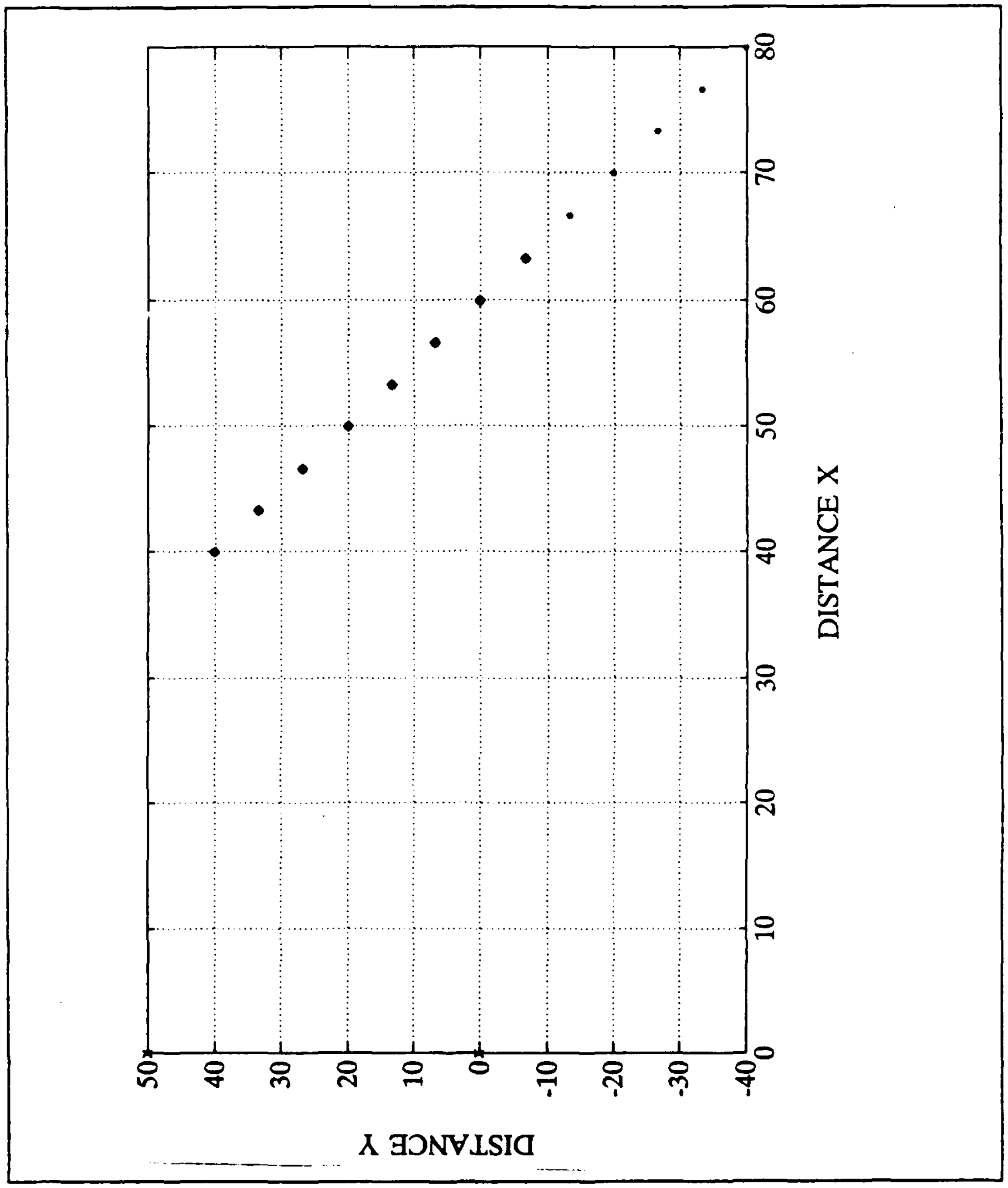

Figure 16. **Actual, Estimated, and Predicted Tracks:** Target-Bearing and Range Observables. Actual Track - o, Estimated Track - +, Predicted Track - *.

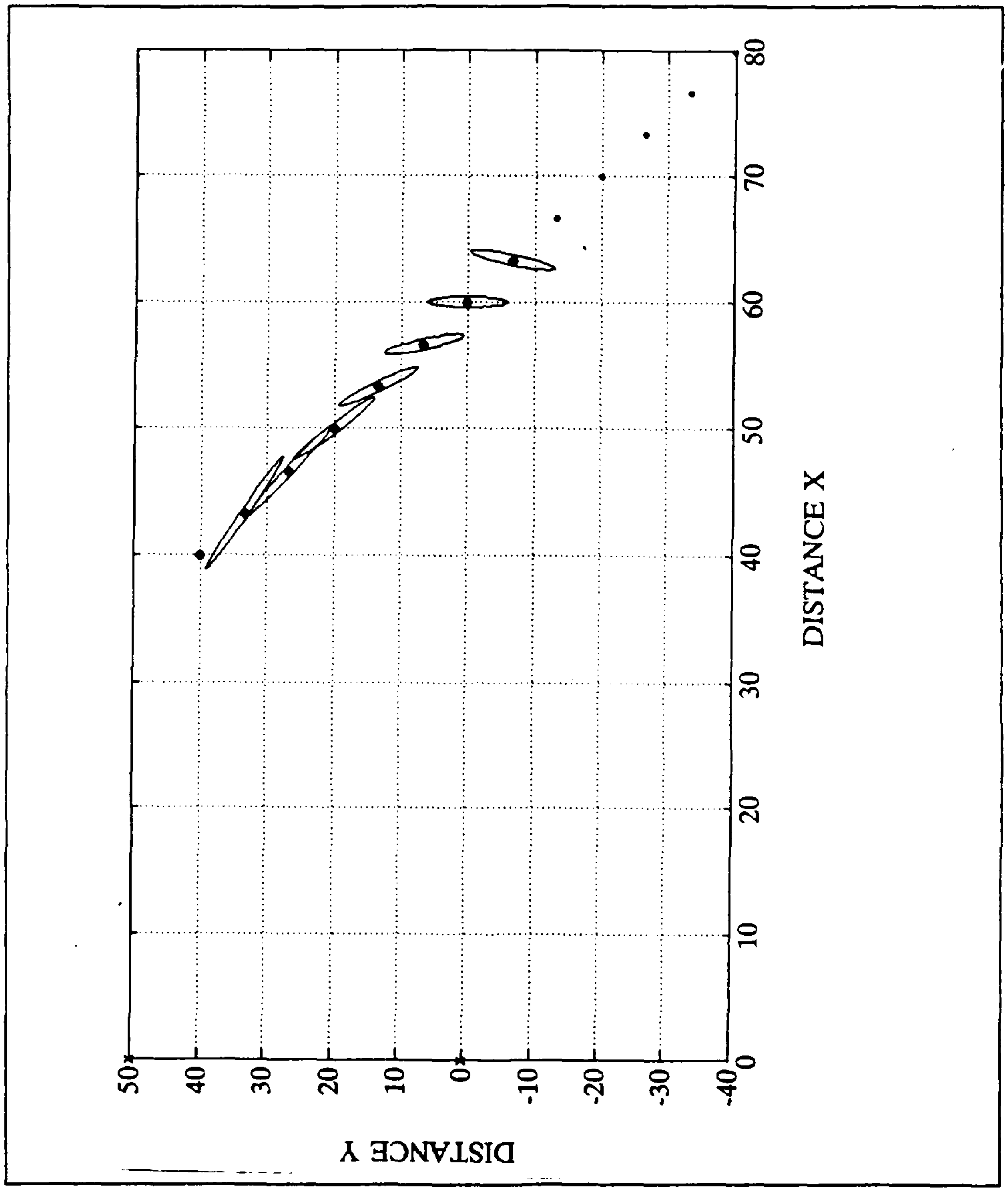

Figure 17. Total Target Track View: Target-Bearing and Range Observables. Actual Track - o, Observed Track - • , Sensor - x. Predicted Track - *.

4. Doppler-Frequency Observable

The results of a problem where a Doppler-shifted frequency and the associated bearing to that frequency are employed are shown in Figure 18 on page 46 through Figure 21 on page 49. These are based on a system employing one sensor, which is stationary and located at the origin. Figure 18 on page 46 shows the actual track positions compared to the target observations. If the observations compare to the actual positions, white noise is added and the observables sent to the Kalman algorithm for processing. Figure 19 on page 47 shows the actual and estimated tracks with the error ellipsoids to aid in defining target position. Figure 20 on page 48 shows the actual, estimated , and predicted target tracks. Figure 21 on page 49 is again a combination of the previous three graphs and would be the one seen in the field. The target movement in this case was from upper left to lower right.

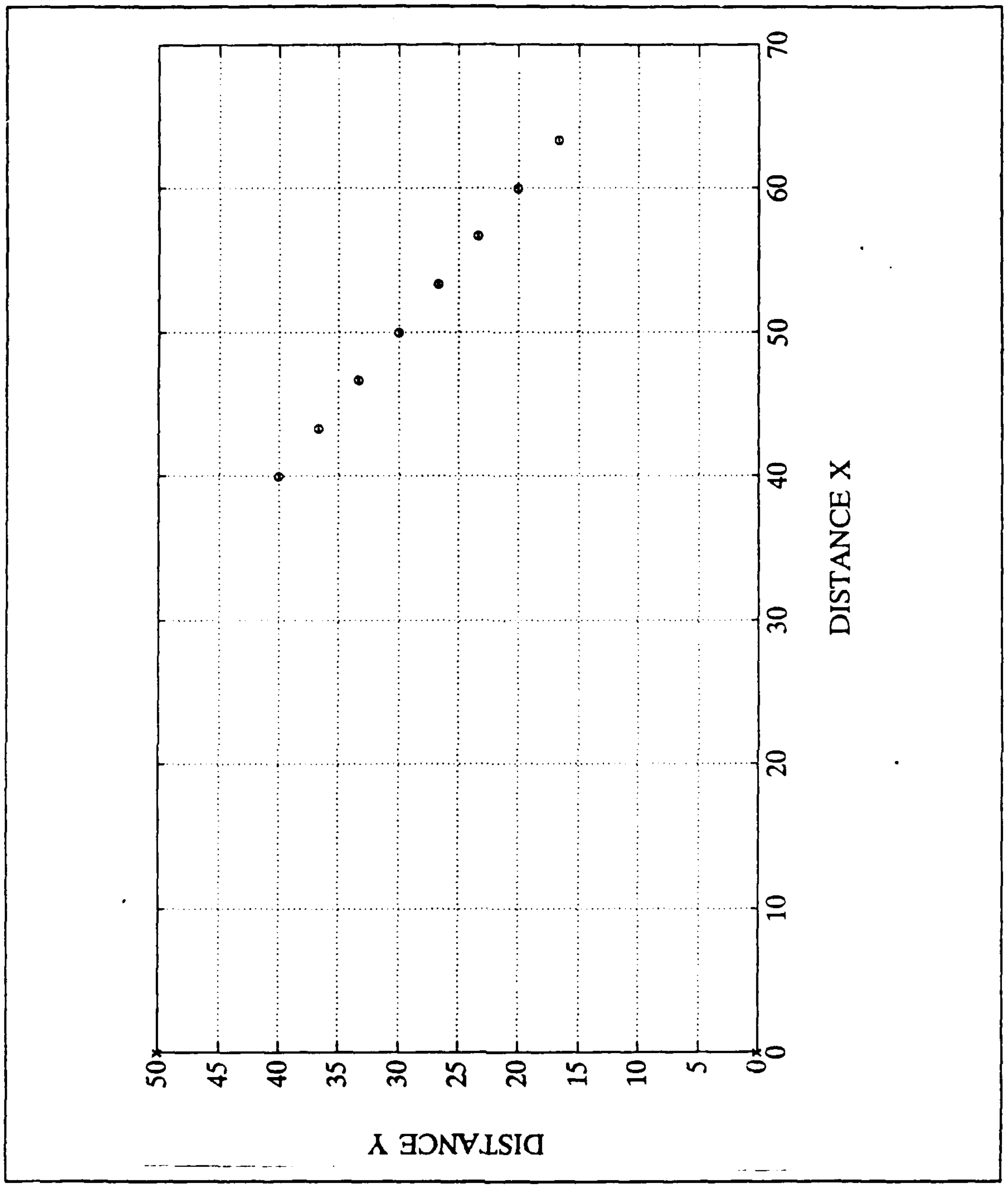

Figure 18. **Actual and Observed Tracks:** Doppler Frequency Observable. Actual Track - o, Observed Track - • , Sensor - x.

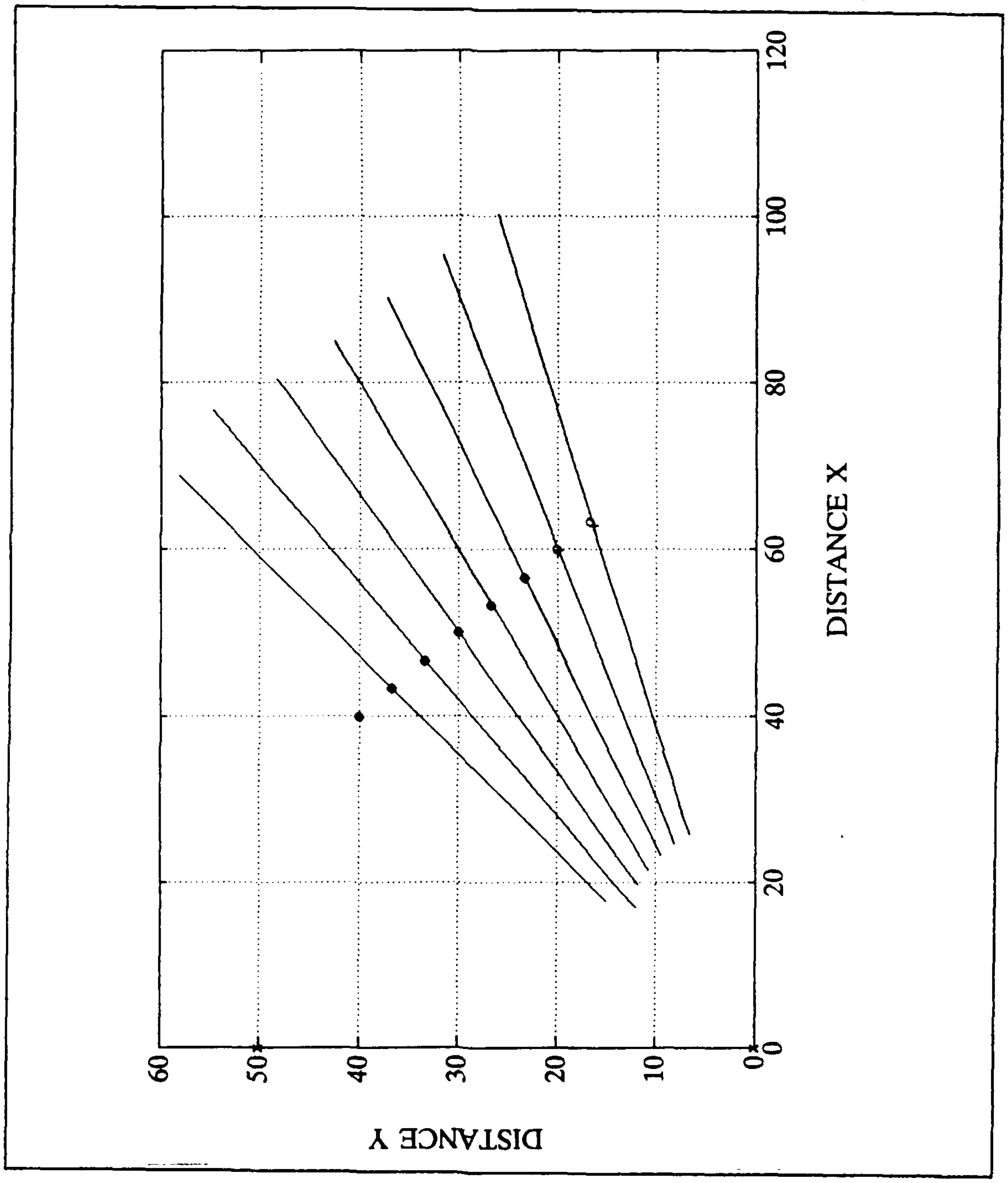

Figure 19. **Actual and Estimated Tracks:** Doppler Frequency Observable. Actual Track - o, Estimated Track - + . The Plot Also Contains Error Ellipsoids to Aid in Target Location.

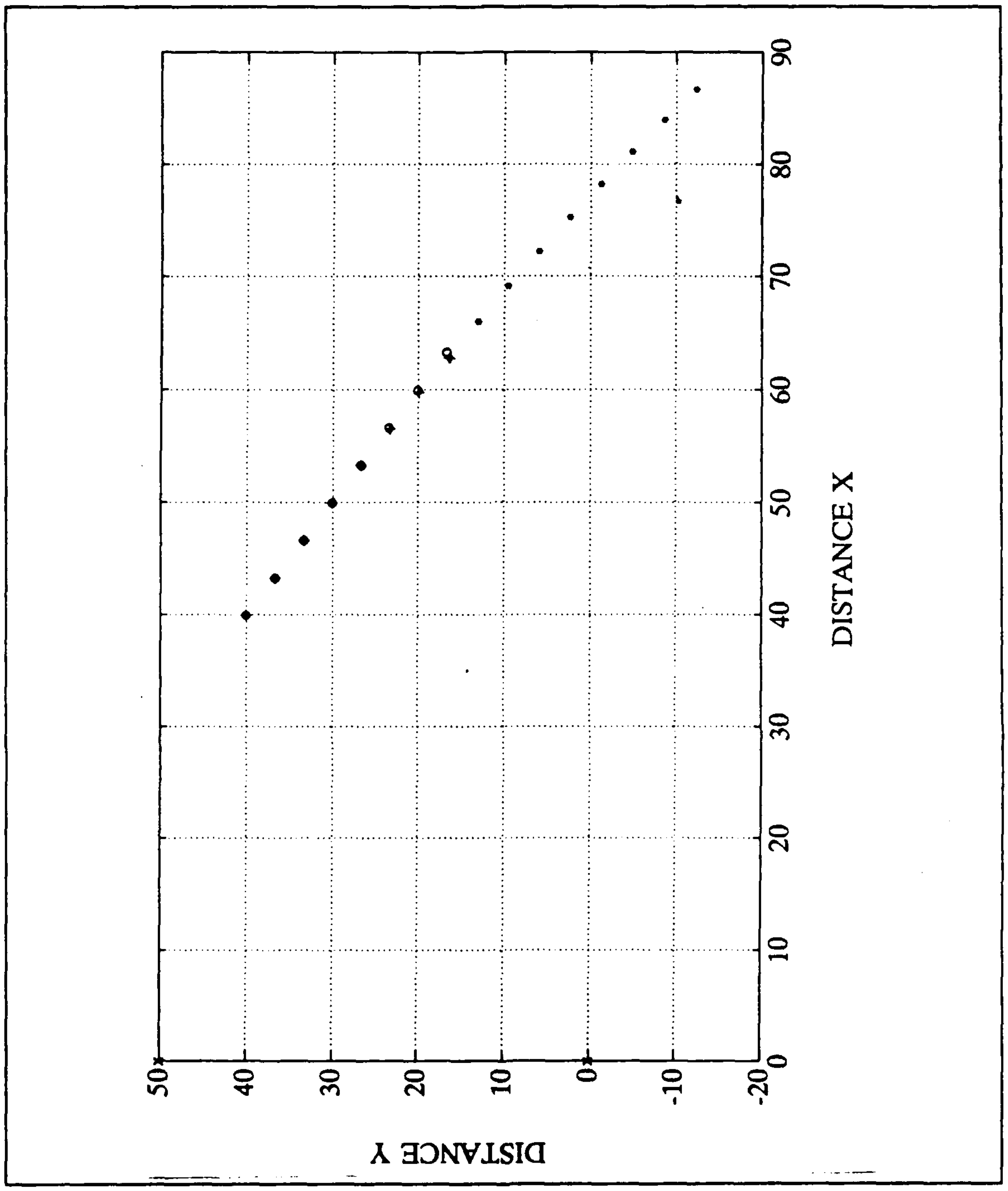

Figure 20. **Actual, Estimated, and Predicted Tracks:** Doppler Frequency Observable. Actual Track - o, Estimated Track - +, Predicted Track - *.

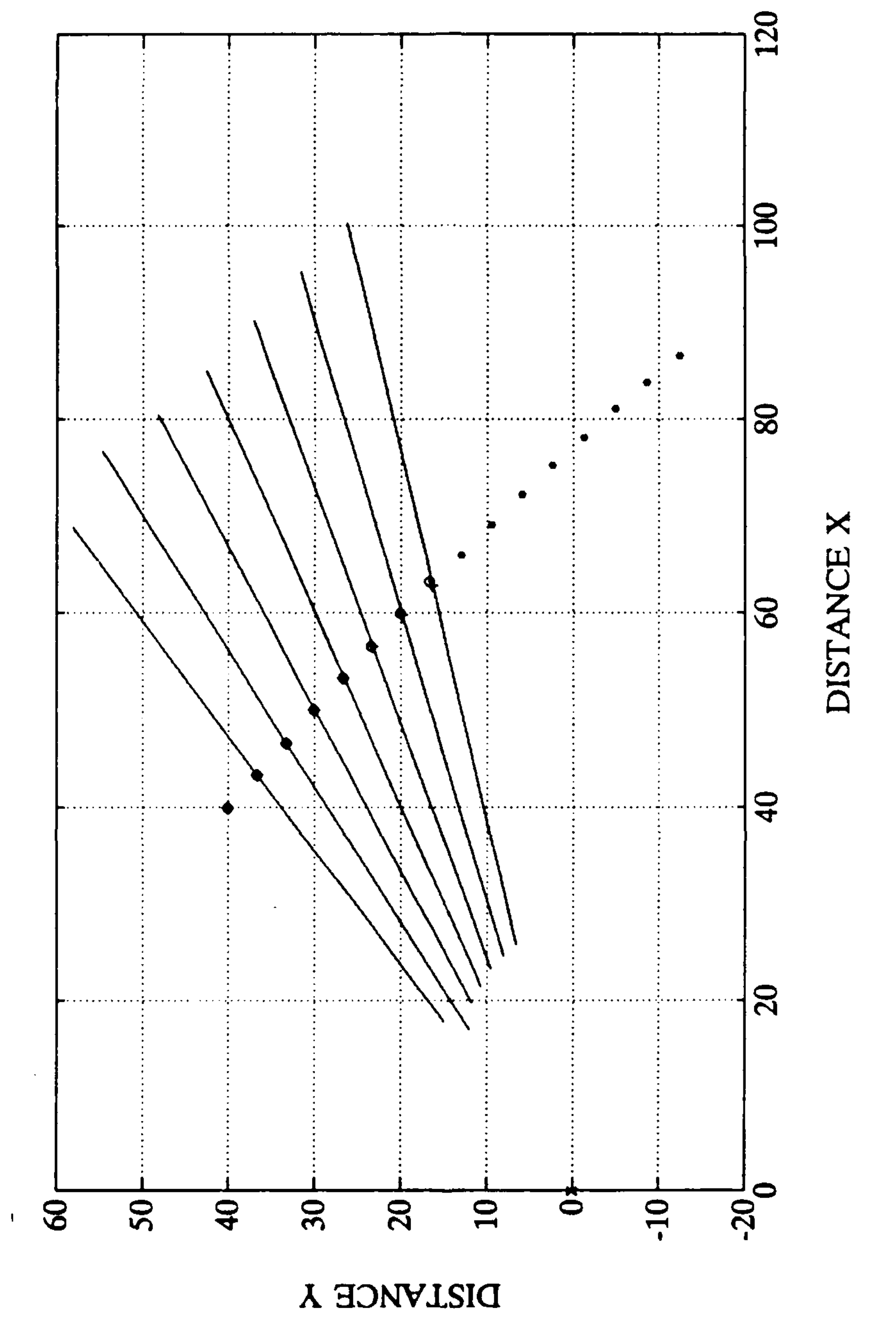

Figure 21. **Total Target Track View:** Doppler Frequency Observable. Actual Track - o, Observed Track - • , Sensor - x. Predicted Track - *.

VII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The target tracking algorithm developed here is both personal computer-based and user-friendly. This would aid in field employment where the system would be of most use to the maneuver commander. With improvements in the tracking algorithm presented, or on one similiarly based, real-time tracking could be accomplished. These improvements will be subsequently discussed. With real-time tracking, indirect supporting arms or direct air could engage the target with a reasonable chance of success. If the orientation of the target were known, optimal damage could then be inflicted upon it. Hence, the development of a working fused-sensor system tracking algorithm would place forward reconnaissance personnel at intelligent risk when seeking combat maneuver intelligence far forward of the FEBA.

B. FUTURE STUDY POSSIBILITIES

Future study regarding improvements in the algorithm developed is quite extensive. Topics which would facilitate the program's usefulness include incorporation of the acceleration terms in the X and Y *directions* and the application of the Z coordinate and its velocity and acceleration terms. PLRS uses time division multiple access (TDMA) for unit communications transmissions. A delay study should be done so as to account for a worst-case analysis of the effect that transmission delays would have on the employment of the Kalman tracker or its target update capability.

C. PROGRAM IMPROVEMENTS

Improved program structure can be achieved by employment of the principles used by Baheti [Ref. 8]. The Baheti algorithm requires one quarter of the memory capacity and fewer computations than the algorithm as developed here, but does not adhere to conventional control methods. The algorithm herein presented was written in PC Matlab which is a matrix manipulation program. It will require updating as improvements in the Matlab algorithm are developed.

Since optimal tracking of a ground target would be futile in the fact that the ground target motion would be unpredictable, another possible program improvement could be researched. The Defense Mapping Agency has worldwide computer mapping data which can be displayed on a computer terminal. It may be possible to incorporate this data in a logic algorithm structure so as to determine the target's tendencies to augment the optimal tracking scenario. Once the logical choices are made, the potential maneuver commander could employ logic as well as optimal control to thwart his adversary.

APPENDIX TARGET TRACKING SIMULATION PROGRAMS

A. PROGRAM DISCUSSION

The programs contained herein were written to simulate the various aspects of the target tracking problem heretofore discussed. They are written in PC-Matlab and hence, are denoted by a .m suffix to identify them as control or execution files. A caveat should be issued at this point. This set of programs was written in PC-Matlab version 3.5 which contains some newer function files not previously available. One should check the version of PC-Matlab employed before attempting tracking simulations. If the function files necessary to run these programs are not present, an error will result. The programs are commented throughout in the hope that they will aid the user in understanding their operation.

The main driver file for the set is titled marine.m. The subroutines are titled brgbrg.m, rngrng.m, brgrng.m, and doppler.m. The file errellip.m is called by each of the subroutines as a function file. The function file reshape.m, is also necessary for the operation of marine.m. It is contained for reference. These programs should not be considered as validated or free of logical or arithmetic error.

B. PROGRAM LISTING

```
%  JOHN A. HUCKS II
%  FILE NAME: MARINE.M
%  CONTROL FILE FOR MULTIPLE GEOMETRY EXTENDED KALMAN FILTER TARGET
%  TRACKING SIMULATION PROGRAM.
%  FILES NECESSARY FOR THE OPERATION OF THIS PROGRAM ARE:
%                     BRGBRG.M
%                     RNGRNG.M
%                     BRGRNG.M
%                     DOPPLER.M
%                     ERRELLIP.M       (COURTESY OF STEPHEN L. SPEHN)
%                     RESHAPE.M        (FUNCTION FILE)
%  THIS PROGRAM DRIVES OTHER MATLAB M (EXECUTION) FILES WHICH SIMULATE
%  TARGET TRACKING BASED ON VARIOUS OBSERVABLES, THESE BEING BEARINGS,
%  RANGES, BEARINGS AND RANGES, OR A DOPPLER SHIFTED FREQUENCY. THE
%  PROGRAMS ARE BASED ON FUSED MULTIPLE OBSERVERS THAT ARE EITHER
%  STATIONARY OR MOVING.

%  PROGRAM SETUP

clear,hold off,subplot(111),clg,clc
titleline = '                         TARGET TRACKING SIMULATOR';
titleline1 = '                                  (OPTIMAL)          ';
```

```
titleline2 = '                    NAVAL POSTGRADUATE SCHOOL MSEE THESIS';
titleline3 = '                  CAPTAIN JOHN A. HUCKS II USMC  DEC. 1989';

% TITLE DISPLAY

disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp(titleline);
disp(titleline1);
disp(titleline2);
disp(titleline3);
pause

% OBSERVER INPUT ROUTINE

clc
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp('ENTER THE FIRST OBSERVERS INITIAL PARAMETERS:');
disp(' ');
disp(' ');
disp('          xfo1 = [ x              % observer x position (km) ')
disp('                   vx             % velocity x direction (km/hr)')
disp('                   y              % observer y direction (km)')
disp('                   vy ];          % velocity y direction (km/hr)')
disp(' ');
disp('YOU MAY ENTER THIS IN THE FOLLOWING FORMAT:[x vx y vy]' ');
xfo1s = input('xfo1 = ','s');
eval(['xfo1 = ',xfo1s,';']);
xfo1 = reshape(xfo1,4,1);

clc
disp(' ');
disp(' ');
disp(' ');
disp(' ');
disp('ENTER THE SECOND OBSERVERS INITIAL PARAMETERS:');
disp(' ');
disp(' ');
disp('          xfo2 = [ x            % observer x position (km)')
disp('                   vx           % velocity x direction (km/hr)')
disp('                   y            % observer y direction (km) ')
disp('                   vy ];        % velocity y direction (km/hr)')
disp(' ');
disp('YOU MAY ENTER THIS IN THE FOLLOWING FORMAT:[x vx y vy]'');
```

```
xfo2s = input('xfo2 = ','s');
eval(['xfo2 = ',xfo2s,';']);
xfo2 = reshape(xfo2,4,1);

clc
disp(' ');
disp('ENTER OBSERVER MOVEMENT SPEED:');
disp(' ');
disp('                    spd                   % observer speed in km/hr ')
disp(' ');
disp(' ');
spds = input('spd = ','s');
eval(['spd = ',spds,';']);

clc
disp(' ');
disp('ENTER DESIRED NUMBER OF OBSERVATIONS:');
disp(' ');
disp('                    kmax                       % observation number ')
disp(' ');
disp(' ');
disp('PLEASE MAKE THIS GREATER THAN 5 SO THAT THE KALMAN GAINS')
disp('CAN STABILIZE.')
kmaxs = input('kmax = ','s');
eval(['kmax = ',kmaxs,';']);

clc
disp(' ');
disp('ENTER DESIRED NUMBER OF PREDICTIONS:');
disp(' ');
disp('                    kpred                      % prediction number ')
disp(' ');
disp(' ');
kpreds = input('kpred = ','s');
eval(['kpred = ',kpreds,';']);

clc
disp(' ');
disp('ENTER OBSERVATION INTERVAL:');
disp(' ');
disp('                    dt                        % observation interval')
disp(' ');
disp('THE TIME INTERVAL IS IN THE FORM:')
disp(' ')
disp('                          dt = time/60')
disp(' ');
disp('FOR A 10 MINUTE INTERVAL ENTER:  dt = 10/60')
dts = input('dt = ','s');
eval(['dt = ',dts,';']);

%  TARGET SIMULATION PARAMETERS INPUT ROUTINE

clc
disp(' ');
disp(' ');
disp('       FOR THE PURPOSES OF SIMULATION,');
disp('          ENTER THE TARGETS MOVEMENT PARAMETERS:');
disp(' ');
disp(' ');
disp('                    [ x      % target initial x position (km)')
disp('                      vx     % target initial x velocity (km/hr)')
disp('                      ax     % target acceleration in x (km/hr^2)')
disp('                      y      % target initial y position (km)')
disp('                      vy     % target initial y velocity (km/hr)')
disp('                      ay     % target acceleration in y (km/hr^2)')
disp('                      fo ];  % transmitter rest frequency (Hz)')
disp(' ');
disp(' ');
disp('YOU MAY ENTER THIS IN THE FOLLOWING FORMAT:[x vx ax y vy ay fo]''');
disp(' ');
disp(' ');
disp(' ****************************************************  ')
disp(' * CAUTION: VELOCITY OF THE TARGET MUST NOT BE ZERO. *  ')
disp(' ****************************************************  ')
tgts = input('tgt = ','s');
eval(c'tgt = ',tgts,';'|);
tgt = reshape(tgt,7,1);
if (tgt(2,1) == 0 & tgt(5,1) == 0)
    clc
    disp(' ');
    disp(' ');
    disp('          FOR THE PURPOSES OF SIMULATION,');
    disp('          ENTER THE TARGETS MOVEMENT PARAMETERS:');
    disp(' ');
    disp(' ');
    disp('                [ x          % target initial x position (km)')
    disp('                  vx         % target initial x velocity (km/hr)')
    disp('                  ax         % target acceleration in x (km/hr^2)')
    disp('                  y          % target initial y position (km)')
    disp('                  vy         % target initial y velocity (km/hr)')
    disp('                  ay         % target acceleration in y (km/hr^2)')
    disp('                  fo ];      % transmitter rest frequency (Hz)')
    disp(' ');
    disp(' ');
    disp('YOU MAY ENTER THIS IN THE FOLLOWING FORMAT:
    [x vx ax y vy ay fo]''');
    disp(' ');
    disp(' ');
    disp(' ****************************************************  ')
    disp(' * CAUTION: VELOCITY OF THE TARGET MUST NOT BE ZERO. *  ')
    disp(' ****************************************************  ')
    tgts = input('tgt = ','s');
    eval(['tgt = ',tgts,';']);
    tgt = reshape(tgt,7,1);
end

global xfo1 xfo2 spd kmax kpred dt tgt;

% OBSERVABLE MENU CHOICE ROUTINE

choice1 = -10;
while choice1  = 0
```

```
clc
disp(' ');
disp(' ');
disp(' ');
disp(titleline);
disp(titleline1);
disp(' ');
disp(' ');
disp(' ');
disp('          ENTER THE METHOD OF OBSERVATION USED FOR THIS RUN:');
disp(' ');
disp('                    1. Bearing/Bearing');
disp('                    2. Bearing/Range');
disp('                    3. Range/Range');
disp('                    4. Shifted Doppler Frequency');
disp('                    5. Display the last graph');
disp('                    6. Change the number of observations');
disp('                    7. Change the number of predictions');
disp('                    8. Change the observation interval');
disp('                    9. Change position of first observer');
disp('                    10. Change position of second observer');
disp('                    11. Change observer movement speed');
disp('                    12. Change target simulation parameters');
disp(' ');
disp('                    0. Quit');
disp(' ');
choice1 = input('           ENTER YOUR CHOICE: ');
if choice1 == 1
   brgbrg;
elseif choice1 == 2
   brgrng;
elseif choice1 == 3
   rngrng;
elseif choice1 == 4
   doppler;
elseif choice1 == 5
   shg
   pause
elseif choice1 == 6
    clc
    disp(' ');
    disp('ENTER DESIRED NUMBER OF OBSERVATIONS:');
    disp(' ');
    disp('                 kmax          % observation number ')
    disp(' ');
    disp(' ');
    disp('PLEASE MAKE THIS GREATER THAN 5 SO THAT THE KALMAN GAINS')
    disp('CAN STABILIZE.')
    kmaxs = input('kmax = ','s');
    eval(['kmax = ',kmaxs,';']);
elseif choice1 == 7
    clc
    disp(' ');
    disp('ENTER DESIRED NUMBER OF PREDICTIONS:');
    disp(' ');
    disp('                 kpred         % prediction number ')
    disp(' ');
    disp(' ');
    kpreds = input('kpred = ','s');
    eval(['kpred = ',kpreds,';']);
elseif choicel == 8
    clc
    disp(' ');
    disp('ENTER OBSERVATION INTERVAL:');
    disp(' ')
    disp('                    dt                  % observation interval')
    disp(' ');
    disp('THE TIME INTERVAL IS IN THE FORM:')
    disp(' ')
    disp('                          dt = time/60')
    disp(' ');
    disp('FOR A 10 MINUTE INTERVAL ENTER:  dt = 10/60')
    disp(' ');
    dts = input('dt = ','s');
    eval(['dt = ',dts,';']);
elseif choicel == 9
    clc
    disp(' ');
    disp(' ');
    disp(' ');
    disp(' ');
    disp('ENTER THE FIRST OBSERVERS INITIAL PARAMETERS:');
    disp(' ');
    disp(' ');
    disp('     xfo1 = [ x          % observer x position (km) ')
    disp('              vx         % velocity x direction (km/hr) ')
    disp('              y          % observer y direction (km)')
    disp('              vy ];      % velocity y direction (km/hr)')
    disp(' ');
    disp('YOU MAY ENTER THIS IN THE FOLLOWING FORMAT:[x vx y vy]''');
    xfo1s = input('xfo1 = ','s');
    eval(['xfo1 = ',xfo1s,';']);
    xfo1 = reshape(xfo1,4,1);
elseif choicel == 10
    clc
    disp(' ');
    disp(' ');
    disp(' ');
    disp(' ');
    disp('ENTER THE SECOND OBSERVERS INITIAL PARAMETERS:');
    disp(' ');
    disp(' ');
    disp('     xfo2 = [ x          % observer x position (km)')
    disp('              vx         % velocity x direction (km/hr)')
    disp('              y          % observer y direction (km)')
    disp('              vy ];      % velocity y direction (km/hr)')
    disp(' ');
    disp('YOU MAY ENTER THIS IN THE FOLLOWING FORMAT:[x vx y vy]''');
    xfo2s = input('xfo2 = ','s');
    eval(['xfo2 = ',xfo2s,';']);
    xfo2 = reshape(xfo2,4,1);
elseif choicel == 11
```

```
        clc
        disp(' ');
        disp('ENTER OBSERVER MOVEMENT SPEED:');
        disp(' ');
        disp('              spd              % observer speed in km/hr ')
        disp(' ');
        disp(' ');
        spds = input('spd = ','s');
        eval(['spd = ',spds,';']);
    elseif choice1 == 12
        clc
        disp(' ');
        disp(' ');
        disp('          FOR THE PURPOSES OF SIMULATION,');
        disp('          ENTER THE TARGETS MOVEMENT PARAMETERS:');
        disp(' ');
        disp(' ');
        disp('           [ x          % target initial x position (km)')
        disp('             vx         % target initial x velocity (km/hr)')
        disp('             ax         % target acceleration in x (km/hr^2)')
        disp('             y          % target initial y position (km)')
        disp('             vy         % target initial y velocity (km/hr)')
        disp('             ay         % target acceleration in y (km/hr^2)')
        disp('             fo ];      % transmitter rest frequency (Hz)')
        disp(' ');
        disp(' ');
        disp('YOU MAY ENTER THIS IN THE FOLLOWING FORMAT:
        [x vx ax y vy ay fo]'')
;
        disp(' ');
        disp(' ');
        disp(' *****************************************************  ')
        disp(' * CAUTION: VELOCITY OF THE TARGET MUST NOT BE ZERO. *  ')
        disp(' *****************************************************  ')
     tgts = input('tgt = ','s');
     eval(['tgt = ',tgts,';']);
     tgt = reshape(tgt,7,1);
        if (tgt(2,1) == 0 & tgt(5,1) == 0)
            clc
            disp(' ');
            disp(' ');
            disp('          FOR THE PURPOSES OF SIMULATION,');
            disp('          ENTER THE TARGETS MOVEMENT PARAMETERS:');
            disp(' ');
            disp(' ');
            disp('       [ x      % target initial x position (km)')
            disp('         vx     % target initial x velocity (km/hr)')
            disp('         ax     % target acceleration in x (km/hr^2)')
            disp('         y      % target initial y position (km)')
            disp('         vy     % target initial y velocity (km/hr)')
            disp('         ay     % target acceleration in y (km/hr^2)')
            disp('         fo ];  % transmitter rest frequency (Hz)')
            disp(' ');
            disp(' ');
            disp('YOU MAY ENTER THIS IN THE FOLLOWING FORMAT:
            [x vx ax y vy ay fo]'');
```

```
                disp(' ');
                disp(' ');
                disp(' ***********************************************************')
                disp(' * CAUTION: VELOCITY OF THE TARGET MUST NOT BE ZERO. *')
                disp(' ***********************************************************')
                tgts = input('tgt = ','s');
                eval(['tgt = ',tgts,';']);
                tgt = reshape(tgt,7,1);
            end
        end;
    end;


% JOHN A. HUCKS II
% FILE NAME : BRGBRG.M
% EXTENDED KALMAN FILTER PROGRAM USING BEARING/BEARING MEASUREMENTS

% THIS PROGRAM SIMULATES THE TRACK GENERATED BY AN EXTENDED KALMAN
% FILTER, WITH INPUTS FROM A FUSED SENSOR ARRAY AS IT TRACKS A TARGET
% ACROSS AN X-Y GRID SPACE (PLRS FLAT EARTH PROJECTION).

clg
clear eex eey xto
!erase brgbrg.met
diary bearing

%  STATE EQUATIONS

%  OBSERVATION TIME INTERVAL (input from marine.m)

%  STATE TRANSITION MATRIX
phi = [ 1 dt 0  0
        0  1 0  0
        0  0 1 dt
        0  0 0  1 ];

%  SYSTEM NOISE COEFFICIENT MATRIX
del = [ dt*dt/2    0
          dt       0
          0      dt*dt/2
          0         dt    ];

%  STATE VARIABLES, FILTER ESTIMATES, TIME, AND OBSERVED DATA
%  NUMBER OF OBSERVATIONS (input from marine.m)


%  ESTIMATED TARGET STATE
xte  = zeros(4,kmax);                    % initial est. target state

%  OBSERVER (SENSOR) INITIAL POSITIONS AND VELOCITIES
%  (input from marine.m)

%  OBSERVER SPEED (input from marine.m)
% INITIAL TIME SETTING
Time = zeros(1,kmax);

% ACTUAL TRACK POSITIONS  (tgt vector input from marine.m)
x = zeros(1,kmax);
y = zeros(1,kmax);
T = 0;
for k = 1
        x(k) = tgt(1,1);
        y(k) = tgt(4,1);
end
for k = 2:kmax
        T = T+dt;
        x(k) = (0.5*tgt(3,1)*(T^2))+(tgt(2,1).*T)+(tgt(1,1));
        y(k) = (0.5*tgt(6,1)*(T^2))+(tgt(5,1).*T)+(tgt(4,1));
end
        zpos = [ x
                 y ];

% INPUT OBSERVATION VALUES (BEARING/BEARING)
% INITIAL OBSERVED TARGET STATE
   xto(:,1) = [ tgt(1,1)                 % (km in x direction)
                tgt(2,1)                 % (km/hr in x direction)
                tgt(4,1)                 % (km in y direction)
                tgt(5,1) ];              % (km/hr in y direction)

% SIMULATION COUNTER LOOP
xfoa = zeros(4,kmax);
xfoa(:,1) = xfo1(:,1);
xfo1 = xfoa;
xfob = zeros(4,kmax);
xfob(:,1) = xfo2(:,1);
xfo2 = xfob;
k = 0;
for k1=1:10*kmax
   k = k + 1;                          % actual loop counter
    if (k == kmax)                     % check for termination
      break
    end

% UPDATE FORWARD OBSERVER'S/ FAC'S POSITION (IF THEY ARE MOVING)
    if  k <= 3
      way = pi/2;
    else
      way = -(3*pi/4);
    end
    if spd == 0
      xfo1(1,k+1) = xfo1(1,k);
      xfo1(3,k+1) = xfo1(3,k);
      xfo2(1,k+1) = xfo2(1,k);
      xfo2(3,k+1) = xfo2(3,k);
    else
      xfo1(1,k+1) = xfo1(1,k) + spd*sin(way)*dt;
      xfo1(2,k+1) = spd*sin(way);
      xfo1(3,k+1) = xfo1(3,k) + spd*cos(way)*dt;
      xfo1(4,k+1) = spd*cos(way);
```

```
        xfo2(1,k+1) = xfo2(1,k) + spd*sin(way)*dt;
        xfo2(2,k+1) = spd*sin(way);
        xfo2(3,k+1) = xfo2(3,k) + spd*cos(way)*dt;
        xfo2(4,k+1) = spd*cos(way);
      end
    end
    delt1x = zpos(1,:) - xfo1(1,:);
    delt1y = zpos(2,:) - xfo1(3,:);
    delt2x = zpos(1,:) - xfo2(1,:);
    delt2y = zpos(2,:) - xfo2(3,:);
    brg1 =  atan2(delt1x,delt1y);            % bearing thta1
    brg2 =  atan2(delt2x,delt2y);            % bearing thta2
    zobs = [brg1;brg2];
    xto1 = zeros(4,kmax);
    xto1(1,:) = zpos(1,:);
    xto1(3,:) = zpos(2,:);
    xto1(:,1) = xto(:,1);
    xto = xto1;

% STATE EXCITATION AND MEASUREMENT ERROR COVARIANCE MATRICES
varv  = 0.0001;                       % variance of linear acceleration

varth = 0.01096;                      % variance of angular acceleration
R  = [ 0.000005  0
        0    0.000005 ];              % measurement noise

% INPUT OBSERVATION VALUES WITH NOISE ADDED
rand('normal')
zobs = zobs + sqrt(R(1,1))*rand(zobs);

% INITIAL KALMAN MATRICES AND OBSERVATION
I = eye(4);

% INITIAL KALMAN P MATRIX
P0 = [ 800   0  0    0                % reset value of Error Covariance
       0    800 0    0
       0     0  800  0
       0     0  0   800 ];

% REINITIALIZING P MATRIX
P = P0;

% KALMAN ESTIMATE
xte(:,1) = [ xto(1,1)                      % (km - x direction)
             tgt(2,1)                      % (km/hr)
             xto(3,1)                      % (km - y direction)
             tgt(5,1) ];                   % (km/hr)

tpred = xte(:,1);

% SIMULATION OF KALMAN FILTER OPERATION
% NUMBER OF BAD OBSERVATIONS
nbad = 0;                                  % initial setting bad obs. cntr.
% SIMULATION COUNTER LOOP
k = 0;
for k1=1:10*kmax
```

```
   k = k + 1;                                  % actual loop counter
   if (k == kmax)                              % check for termination
      break
   end

%  UPDATE FORWARD OBSERVER'S/ FAC'S POSITION (IF THEY ARE MOVING)
   if  k <= 3
      way = pi/2;
   else
      way = -(3*pi/4);
   end
   if spd == 0
      xfo1(1,k+1) = xfo1(1,k);
      xfo1(3,k+1) = xfo1(3,k);
      xfo2(1,k+1) = xfo2(1,k);
      xfo2(3,k+1) = xfo2(3,k);
   else
      xfo1(1,k+1) = xfo1(1,k) + spd*sin(way)*dt;
      xfo1(2,k+1) = spd*sin(way);
      xfo1(3,k+1) = xfo1(3,k) + spd*cos(way)*dt;
      xfo1(4,k+1) = spd*cos(way);
      xfo2(1,k+1) = xfo2(1,k) + spd*sin(way)*dt;
      xfo2(2,k+1) = spd*sin(way);
      xfo2(3,k+1) = xfo2(3,k) + spd*cos(way)*dt;
      xfo2(4,k+1) = spd*cos(way);
   end

   %  PROJECT AHEAD KALMAN STATE ESTIMATE
   xte(: ,k+1) = phi*xte(: ,k);

   %  UPDATE Q
   vt = sqrt((xte(2,k+1))^2 + (xte(4,k+1))^2);
   q11 = varv * (xte(2,k+1) / vt)^2 + varth * (xte(4,k+1))^2;
   q22 = varv * (xte(4,k+1) / vt)^2 + varth * (xte(2,k+1))^2;
   q12 = xte(2,k+1) * xte(4,k+1) * ((varv/vt)^2 - varth );
   q21 = q12;
   Q1 = [  q11 q12
           q21 q22 ];
   Q = del*Q1*(del');

   %  PROJECT ERROR COVARIANCE
   P = phi*P*(phi') + Q;

   %  UPDATE H
   delt1x = xte(1,k+1) - xfo1(1,k+1);
   delt1y = xte(3,k+1) - xfo1(3,k+1);
   delt2x = xte(1,k+1) - xfo2(1,k+1);
   delt2y = xte(3,k+1) - xfo2(3,k+1);
   r12 = (delt1x) 2 + (delt1y) 2;
   r22 = (delt2x) 2 + (delt2y) 2;
   r1 = sqrt(r12);
   r2 = sqrt(r22);
   H = [  delt1y/r12  0   -delt1x/r12  0
          delt2y/r22  0   -delt2x/r22  0 ];

   %  GET NEW MEASURED ESTIMATE
```

```
zhat1 =  atan2(delt1x,delt1y);                % bearing estimate thta1
zhat2 =  atan2(delt2x,delt2y);                % bearing estimate thta2
zhat  = [ zhat1
          zhat2 ];

%  COMPUTATION OF KALMAN GAIN
vresid = (H*P*(H') + R);                  % variance of the residual
G = P*(H') / vresid;                      % Kalman gain

%  COMPUTATION OF UPDATED ERROR COVARIANCE
P = (I - G*H)*P;

%  COMPUTATION OF RESIDUAL
resid = zobs(:,k+1) - zhat;

%  IF RESIDUAL IS TOO BIG, THEN RESET P AND DO CALCULATIONS AGAIN.
if (abs(resid(1)) > 1.0*sqrt(abs(vresid(1,1))) ...
      ] abs(resid(2)) > 1.0*sqrt(abs(vresid(2,2))) ...
      & nbad == 0)
   nbad = 1;
   k = k-1;
   P = P0;                                % reset P
else
%  COMPUTATION OF UPDATED ESTIMATE
   xte(:,k+1) = xte(:,k+1) + G*resid;

%  ERROR ELLIPSOID PLOTTING ROUTINE
   M = [ xte(1,k+1) xte(3,k+1) ]';
   K = [ P(1,1) P(1,3)
         P(3,1) P(3,3) ];
   [xp,yp] = errellip(M,K,.60,50,0);
   eex(:,k) = xp;
   eey(:,k) = yp;

%  PREDICTED (FUTURE TRACK) USING KALMAN ESTIMATE
%  (number of predictions desired input from marine.m)
   tpred(:,1) = xte(:,k+1);
   acc = zeros(2,1);
   for l = 1:kpred                          % kpred is # of predictions
       if k >= 3
          accx = (xte(2,k+1)-xte(2,k))/dt;
          accy = (xte(4,k+1)-xte(4,k))/dt;
          acc = [accx;accy];
          if accx >= 5
             break
             acc = zeros(2,1);
          elseif accy >= 5
             break
             acc = zeros(2,1);
          end
       end
       tpred(:,l+1) = phi*tpred(:,l)+del*acc;
   end
   nbad = 0;
end
%  PLOT OBSERVED AND ESTIMATED TRACKS (TARGET AND OBSERVER)
```

```
   plot(xfo1(1,1:k+1),xfo1(3,1:k+1),'xw',xto(1,1:k+1),xto(3,1:k+1)
   ,'.b',...
   xte(1,1:k+1),xte(3,1:k+1),'+w',xfo2(1,1:k+1),xfo2(3,1:k+1),'xw',...
   zpos(1,1:k+1),zpos(2,1:k+1),'ow',tpred(1,:),tpred(3,:),'*g',...
   eex,eey,'-r'),grid
   title('ACTUAL/OBSERVED/ESTIMATED/PREDICTED TGT TRKS USING BRG/BRG ')
   ,...
   xlabel('DISTANCE X'),ylabel('DISTANCE Y')
end
meta brgbrg




% JOHN A. HUCKS II
% FILE TITLE: RNGRNG.M
% EXTENDED KALMAN FILTER PROGRAM USING RANGE/RANGE MEASUREMENTS
%
% THIS PROGRAM SIMULATES THE TRACK GENERATED BY AN EXTENDED KALMAN
% FILTER, WITH INPUTS FROM A FUSED SENSOR ARRAY AS IT TRACKS A TARGET
% ACROSS AN X-Y GRID SPACE (PLRS FLAT EARTH PROJECTION).

clg
clear eex eey xto
!erase rngrng.met
diary rngrng

%  STATE EQUATIONS

%  OBSERVATION TIME INTERVAL (input from marine.m)

%  STATE TRANSITION MATRIX
phi = [  1 dt 0  0
         0  1 0  0
         0  0 1 dt
         0  0 0  1 ];

%  SYSTEM NOISE COEFFICIENT MATRIX
del = [ dt*dt/2     0
          dt        0
          0       dt*dt/2
          0         dt    ];

%  STATE VARIABLES, FILTER ESTIMATES, TIME, AND OBSERVED DATA

%  NUMBER OF OBSERVATIONS (input from marine.m)

%  ESTIMATED TARGET STATE
xte  = zeros(4,kmax);                    % initial est. target state

%  OBSERVER (SENSOR) INITIAL POSITIONS AND VELOCITIES
%  (input from marine.m)

%  OBSERVER SPEED (input from marine.m)
% INITIAL TIME SETTING
Time = zeros(1,kmax);

% ACTUAL TRACK POSITIONS (tgt vector input from marine.m)
x = zeros(1,kmax);
y = zeros(1,kmax);
T = 0;
for k = 1
        x(k) = tgt(1,1);
        y(k) = tgt(4,1);
end
for k = 2:kmax
        T = T+dt;
        x(k) = (0.5*tgt(3,1)*(T^2))+(tgt(2,1).*T)+(tgt(1,1));
        y(k) = (0.5*tgt(6,1)*(T^2))+(tgt(5,1).*T)+(tgt(4,1));
end
zpos = [ x
         y ];

% INPUT OBSERVATION VALUES (RANGE/RANGE)
% OBSERVED TARGET STATE
xfoa = zeros(4,kmax);
xfoa(:,1) = xfo1(:,1);
xfo1 = xfoa;
xfob = zeros(4,kmax);
xfob(:,1) = xfo2(:,1);
xfo2 = xfob;
xto(:,1) = [ tgt(1,1)                      % (km in x direction)
             tgt(2,1)                      % (km/hr in x direction)
             tgt(4,1)                      % (km in y direction)
             tgt(5,1) ];                   % (km/hr in y direction)
% SIMULATION COUNTER LOOP
k = 0;
for k1=1:10*kmax
   k = k + 1;                            % actual loop counter
   if (k == kmax)                        % check for termination
      break
   end

% UPDATE FORWARD OBSERVER'S/ FAC'S POSITION (IF THEY ARE MOVING)
   if  k <= 3
      way = pi/2;
   else
      way = -(3*pi/4);
   end
   if spd == 0
      xfo1(1,k+1) = xfo1(1,k);
      xfo1(3,k+1) = xfo1(3,k);
      xfo2(1,k+1) = xfo2(1,k);
      xfo2(3,k+1) = xfo2(3,k);
   else
      xfo1(1,k+1) = xfo1(1,k) + spd*sin(way)*10*dt;
      xfo1(2,k+1) = spd*sin(way);
      xfo1(3,k+1) = xfo1(3,k) + spd*cos(way)*10*dt;
      xfo1(4,k+1) = spd*cos(way);
      xfo2(1,k+1) = xfo2(1,k) + spd*sin(way)*10*dt;
```

```
        xfo2(2,k+1) = spd*sin(way);
        xfo2(3,k+1) = xfo2(3,k) + spd*cos(way)*10*dt;
        xfo2(4,k+1) = spd*cos(way);
   end
end
r1 = sqrt((zpos(1,:)-xfo1(1,:)).^2 + (zpos(2,:)-xfo1(3,:)).^2);
r2 = sqrt((zpos(1,:)-xfo2(1,:)).^2 + (zpos(2,:)-xfo2(3,:)).^2);
zobs=[r1;r2];
xto1 = zeros(4,kmax);
xto1(1,:) = zpos(1,:);
xto1(3,:) = zpos(2,:);
xto1(:,1) = xto(:,1);
xto = xto1;

%  STATE EXCITATION AND MEASUREMENT ERROR COVARIANCE MATRICES
varv  = 5.0;                           % variance of linear acceleration

varth = 1.0;                           % variance of angular acceleration
R  = [ 0.005  0
         0  0.005 ];                   % measurement noise

%  INPUT OBSERVATION VALUES WITH NOISE ADDED
rand('normal');
zobs = zobs + sqrt(R(1,1))*rand(zobs);

%  INITIAL KALMAN MATRICES AND OBSERVATION
I = eye(4);
%  INITIAL KALMAN P MATRIX
P0 = [ 250   0  0    0                   % reset value of Error Covariance
       0   250 0    0
       0    0  250  0
       0    0   0  250 ];
%  REINITIALIZING P MATRIX
P = P0;
%  KALMAN ESTIMATE
xte(:,1) = [  xto(1,1)                 % (km - x direction)
              tgt(2,1)                 % (km/hr)
              xto(3,1)                 % (km - y direction)
              tgt(5,1) ];              % (km/hr)

tpred = xte(:,1);

%  SIMULATION OF KALMAN FILTER OPERATION
%  NUMBER OF BAD OBSERVATIONS
nbad = 0;                              % initial setting bad obs. cntr.

%  SIMULATION COUNTER LOOP
k = 0;
for k1=1:10*kmax
   k = k + 1;                          % actual loop counter
   if (k == kmax)                      % check for termination
      break
   end

%  UPDATE FORWARD OBSERVER'S/ FAC'S POSITION (IF THEY ARE MOVING)
   if  k <= 3
   way = pi/2;
else
   way = -(3*pi/4);
end
if spd == 0
   xfo1(1,k+1) = xfo1(1,k);
   xfo1(3,k+1) = xfo1(3,k);
   xfo2(1,k+1) = xfo2(1,k);
   xfo2(3,k+1) = xfo2(3,k);
else
   xfo1(1,k+1) = xfo1(1,k) + spd*sin(way)*10*dt;
   xfo1(2,k+1) = spd*sin(way);
   xfo1(3,k+1) = xfo1(3,k) + spd*cos(way)*10*dt;
   xfo1(4,k+1) = spd*cos(way);
   xfo2(1,k+1) = xfo2(1,k) + spd*sin(way)*10*dt;
   xfo2(2,k+1) = spd*sin(way);
   xfo2(3,k+1) = xfo2(3,k) + spd*cos(way)*10*dt;
   xfo2(4,k+1) = spd*cos(way);
end

%  PROJECT AHEAD KALMAN STATE ESTIMATE

xte(:,k+1) = phi*xte(:,k);

%  UPDATE Q
vt = sqrt((xte(2,k+1)) 2 + (xte(4,k+1)) 2);
q11 = varv * (xte(2,k+1) / vt) 2 + varth * (xte(4,k+1)) 2;
q22 = varv * (xte(4,k+1) / vt) 2 + varth * (xte(2,k+1)) 2;
q12 = xte(2,k+1) * xte(4,k+1) * ((varv/vt) 2 - varth );
q21 = q12;
Q1 = [ q11 q12
       q21 q22 ];
Q = del*Q1*(del');

%  PROJECT ERROR COVARIANCE
P = phi*P*(phi') + Q;

%  UPDATE H
delt1x = xte(1,k+1) - xfo1(1,k+1);
delt1y = xte(3,k+1) - xfo1(3,k+1);
delt2x = xte(1,k+1) - xfo2(1,k+1);
delt2y = xte(3,k+1) - xfo2(3,k+1);
r12 = (delt1x) 2 + (delt1y) 2;
r22 = (delt2x) 2 + (delt2y) 2;
r1 = sqrt(r12);
r2 = sqrt(r22);
H = [  delt1x/r1  0   delt1y/r1  0
       delt2x/r2  0   delt2y/r2  0 ];

%  GET NEW MEASURED ESTIMATE
zhat  = [ r1
          r2 ];

%  COMPUTATION OF KALMAN GAIN
vresid = (H*P*(H') + R);            % variance of the residual
G = P*(H') / vresid;                % Kalman gain
```

```
% COMPUTATION OF UPDATED ERROR COVARIANCE
P = (I - G*H)*P;

% COMPUTATION OF RESIDUAL
resid = zobs(:,k+1) - zhat;

% IF RESIDUAL IS TOO BIG, THEN RESET P AND DO CALCULATIONS AGAIN.
if (abs(resid(1)) > 1.0*sqrt(abs(vresid(1,1))) ...
      ] abs(resid(2)) > 1.0*sqrt(abs(vresid(2,2))) ...
      & nbad == 0)
   nbad = 1;
   k = k-1;
   P = P0;                              % reset P
else

% COMPUTATION OF UPDATED ESTIMATE
   xte(:,k+1) = xte(:,k+1) + G*resid;

% ERROR ELLIPSOID PLOTTING ROUTINE
   M = [xte(1,k+1) xte(3,k+1)]';
   K = [ P(1,1) P(1,3)
         P(3,1) P(3,3) ];
   [xp,yp] = errellip(M,K,.60,25,0);
   eex(:,k) = xp;
   eey(:,k) = yp;

% PREDICTED (FUTURE TRACK) USING KALMAN ESTIMATE
% (number of predictions desired input from marine.m)
   tpred(:,1) = xte(:,k+1);
   acc = zeros(2,1);
   for l = 1:kpred                          % kpred is # of predictions
      if k >= 5
        accx = (xte(2,k+1)-xte(2,k))/dt;
        accy = (xte(4,k+1)-xte(4,k))/dt;
        acc = [accx;accy];
        if accx >= 5
           break
           acc = [0;0];
        elseif accy >= 5
           break
           acc = [0;0];
        end
      end
      tpred(:,l+1) = phi*tpred(:,l)+del*acc;
   end
   nbad = 0;
end
% PLOT OBSERVED AND ESTIMATED TRACKS (TARGET AND OBSERVER)
plot(xfo1(1,1:k+1),xfo1(3,1:k+1),'xw',xte(1,1:k+1),
xte(3,1:k+1),'+w',...
xfo2(1,1:k+1),xfo2(3,1:k+1),'xw',zpos(1,1:k+1),zpos(2,1:k+1),'ow',...
xto(1,1:k+1),xto(3,1:k+1),'.b',tpred(1,:),tpred(3,:),'*g',...
eex,eey,'-r'),grid
title('ACTUAL/OBSERVED/ESTIMATED/PREDICTED TGT TRKS
   USING RNG/RNG OBS'),...
   xlabel('DISTANCE X'),ylabel('DISTANCE Y')
end
meta rngrng
diary off



%  JOHN A. HUCKS II
%  FILE NAME : BRGRNG.M
%  EXTENDED KALMAN FILTER USING BEARING/RANGE MEASUREMENTS

%  THIS PROGRAM SIMULATES THE TRACK GENERATED BY AN EXTENDED KALMAN
%  FILTER WITH INPUTS FROM A FUSED SENSOR ARRAY, AS IT TRACKS A
%  POTENTIAL TARGET ACROSS AN X-Y GRID SPACE (PLRS FLAT EARTH
%  PROJECTION).

clg
clear eex eey xto
!erase brgrng.met
diary brgrng

%  STATE EQUATIONS

%  OBSERVATION TIME INTERVAL  (input from marine.m)

%  STATE TRANSISTION MATRIX
phi = [  1 dt 0  0
         0  1 0  0
         0  0 1 dt
         0  0 0  1 ];

%  SYSTEM NOISE COEFFICIENT MATRIX
del = [ dt*dt/2    0
          dt       0
          0      dt*dt/2
          0        dt    ];

%  STATE VARIABLES, FILTER ESTIMATES, TIME, AND OBSERVED DATA

%  NUMBER OF OBSERVATIONS (input from marine.m)


%  ESTIMATED TARGET STATE
xte  = zeros(4,kmax);                    % estimated target state

%  OBSERVER (SENSOR) INITIAL POSITIIONS AND VELOCITIES (input from marine.m)

%  OBSERVER SPEED (input from marine.m)

%  INITIAL TIME SETUP
Time = zeros(1,kmax);

% ACTUAL TRACK POSITIONS (tgt vector input from marine.m)
x = zeros(1,kmax);
```

```
y = zeros(1,kmax);
T = 0;
for k = 1
        x(k) = tgt(1,1);
        y(k) = tgt(4,1);
end
for k = 2:kmax
        T = T+dt;
        x(k) = (0.5*tgt(3,1)*(T^2))+(tgt(2,1)*T)+(tgt(1,1));
        y(k) = (0.5*tgt(6,1)*(T^2))+(tgt(5,1)*T)+(tgt(4,1));
end
        zpos = [ x
                 y ];

% INPUT OBSERVATION VALUES (BEARING/RANGE):
xfoa = zeros(4,kmax);
xfoa(:,1) = xfo1(:,1);
xfo1 = xfoa;
xfob = zeros(4,kmax);
xfob(:,1) = xfo2(:,1);
xfo2 = xfob;
xfor = zeros(4,kmax);
zobr = zeros(1:kmax);
rng1 = zeros(1:kmax);
rng2 = zeros(1:kmax);
delt1x = zeros(1:kmax);
delt2x = zeros(1:kmax);
delt1y = zeros(1:kmax);
delt2y = zeros(1:kmax);
thta1 = zeros(1:kmax);
thta2 = zeros(1:kmax);
for k = 1:kmax
    rng1(k) = sqrt(((zpos(1,k)-xfo1(1,k)).^2) + ...
                   ((zpos(2,k)-xfo1(3,k)).^2));
    rng2(k) = sqrt(((zpos(1,k)-xfo2(1,k)).^2) + ...
                   ((zpos(2,k)-xfo2(3,k)).^2));
    delt1x(k) = zpos(1,k)-xfo1(1,k);
    delt1y(k) = zpos(2,k)-xfo1(3,k);
    delt2x(k) = zpos(1,k)-xfo2(1,k);
    delt2y(k) = zpos(2,k)-xfo2(3,k);
    thta1(k) = atan2(delt1x(k),delt1y(k));
    thta2(k) = atan2(delt2x(k),delt2y(k));
end
zobr = [ thta1;rng1;thta2;rng2 ];

% MEASUREMENT REFERENCE CONVERSION ROUTINE:
for cnt = 1
      xt1 = tgt(1,1);
      xt2 = tgt(1,1);
      yt1 = tgt(4,1);
      yt2 = tgt(4,1);
      xtr = (xt1+xt2)/2;
      ytr = (yt1+yt2)/2;
      delxr = xtr(1,cnt);
      delyr = ytr(1,cnt);
      thtar(1,cnt) = atan2(delxr,delyr);
      rngr(1,cnt) = sqrt((delxr 2)+(delyr^2));
end
for cnt = 2:kmax
      xt1(1,cnt) = zobr(2,cnt)*sin(zobr(1,cnt));
      yt1(1,cnt) = zobr(2,cnt)*cos(zobr(1,cnt));
      xt2(1,cnt) = zobr(4,cnt)*sin(zobr(3,cnt));
      yt2(1,cnt) = zobr(4,cnt)*cos(zobr(3,cnt));
      xtr(1,cnt) = (xt1(1,cnt)+xt2(1,cnt))/2;
      ytr(1,cnt) = (yt1(1,cnt)+yt2(1,cnt))/2;
      delxr = xtr(1,cnt);
      delyr = ytr(1,cnt);
      thtar(1,cnt) = atan2(delxr,delyr);
      rngr(1,cnt) = sqrt((delxr 2)+(delyr^2));
end

% CONVERTED OBSERVATION MATRIX:(CONVERSION OBSERVER TAKEN AS THE ORIGIN)
%  OBSERVED TARGET STATE
xto(:,1) = [ tgt(1,1)                    % (km in x direction)
             tgt(2,1)                    % (km/hr in x direction)
             tgt(4,1)                    % (km in y direction)
             tgt(5,1) ];                 % (km/hr in y direction)
zobs = [ thtar
         rngr ];
xtol = zeros(4,kmax);
xtol(1,:) = zpos(1,:);
xtol(3,:) = zpos(2,:);
xtol(:,1) = xto(:,1);
xto = xtol;

%  STATE EXCITATION AND MEASUREMENT ERROR COVARIANCE MATRICES
varv  = 5.0;                       % variance of linear acceleration
varth = 1.0;                       % variance of angular acceleration
R = [  0.0005  0
       0       0.005 ];            % measurement noise

%  INPUT OBSERVATION VALUES WITH NOISE ADDED
rand('normal')
noise = sqrt(R(1,1))*rand(zobs);

%  INITIAL KALMAN MATRICES AND OBSERVATIONS
I = eye(4);
%  INITIAL KALMAN P MATRIX
P0 = [ 700  0  0  0                  % reset value of error covariance
       0  700  0  0
       0  0  700  0
       0  0  0  700  ];

%  REINITIALIZING P MATRIX
P = P0;

%  INITIAL KALMAN ESTIMATE
xte(:,1) = [ xto(1,1)
             tgt(2,1)
             xto(3,1)
             tgt(5,1) ];
tpred = xte(:,1);
```

```
% SIMULATION OF FUSED SENSOR OPERATION USING EXTENDED KALMAN FILTER
% NUMBER OF BAD OBSERVATIONS
nbad = 0;                               % initial setting bad obs. cntr.
k = 0;
for k1=1:50*kmax
   k = k + 1;                           % actual loop counter
   if (k == kmax)                       % check for termination
      break
   end

   % UPDATE OBSERVER'S/FAC'S POSITION (IF THEY ARE MOVING)
   if  k <= 3
      way = pi/2;
   else
      way = -pi/4;
   end
   if spd == 0
      xfo1(1,k+1) = xfo1(1,k);
      xfo1(3,k+1) = xfo1(3,k);
      xfo2(1,k+1) = xfo2(1,k);
      xfo2(3,k+1) = xfo2(3,k);
   else
      xfo1(1,k+1) = xfo1(1,k) + spd*sin(way)*dt;
      xfo1(2,k+1) = spd*sin(way);
      xfo1(3,k+1) = xfo1(3,k) + spd*cos(way)*dt;
      xfo1(4,k+1) = spd*cos(way);
      xfo2(1,k+1) = xfo2(1,k) + spd*sin(way)*dt;
      xfo2(2,k+1) = spd*sin(way);
      xfo2(3,k+1) = xfo2(3,k) + spd*cos(way)*dt;
      xfo2(4,k+1) = spd*cos(way);
   end

   % PROJECT AHEAD KALMAN STATE ESTIMATE
   xte(:,k+1) = phi*xte(:,k);

   % UPDATE Q
   vt = sqrt((xte(2,k+1))^2 + (xte(4,k+1))^2);
   q11 = varv * (xte(2,k+1) / vt)^2 + varth * (xte(4,k+1))^2;
   q22 = varv * (xte(4,k+1) / vt)^2 + varth * (xte(2,k+1))^2;
   q12 = xte(2,k+1) * xte(4,k+1) * ((varv/vt)^2 - varth );
   q21 = q12;
   Q1 = [ q11 q12
          q21 q22 ];
   Q = del*Q1*(del');

   % PROJECT ERROR COVARIANCE
   P = phi*P*(phi') + Q;

   % UPDATE H
   deltx = xte(1,k+1);
   delty = xte(3,k+1);
   r2 = (deltx) 2 + (delty) 2;
   r = sqrt(r2);
   H = [  delty/r2  0   -deltx/r2  0
          deltx/r   0    delty/r   0 ];
```

```
% GET NEW MEASURED ESTIMATE
delx = xte(1,k+1);
dely = xte(3,k+1);
zhat1 = atan2(delx,dely);
zhat2 = sqrt(delx^2 + dely^2);
zhat = [ zhat1
         zhat2 ];

% COMPUTATION OF KALMAN GAIN
vresid = (H*P*(H') + R);                % variance of the residual
G = P*(H') / vresid;                    % Kalman gain

% COMPUTATION OF PROJECTED ERROR COVARIANCE
P = (I - G*H)*P;

% COMPUTATION OF RESIDUAL
resid = zobs(:,k+1) - zhat;

% IF RESIDUAL TOO BIG, RESET P AND DO CALCULATIONS AGAIN.
if (abs(resid(1)) > 1.0*sqrt(abs(vresid(1,1))) ...
      | abs(resid(2)) > 1.0*sqrt(abs(vresid(2,2))) ...
      & nbad < 2)
   nbad = nbad + 1;
   k = k-1;
   P = P0;                              % reset P
else
   % COMPUTATION OF UPDATED ESTIMATE
   xte(:,k+1) = xte(:,k+1) + G*resid;

   % ERROR ELLIPSOID PLOTTING ROUTINE
   M = [xte(1,k+1) xte(3,k+1)]';
   K = [ P(1,1) P(1,3)
         P(3,1) P(3,3) ];
   [xp,yp] = errellip(M,K,.60,25,0);
   eex(:,k) = xp;
   eey(:,k) = yp;

   % PREDICTED (FUTURE TRACK) USING KALMAN ESTIMATE
   % (number of predictions desired input from marine.m)
   tpred(:,1) = xte(:,k+1);
   acc = zeros(2,1);
   for l = 1:kpred                           % kpred is # of predictions
       if k >= 5
          accx = (xte(2,k+1)-xte(2,k))/dt;
          accy = (xte(4,k+1)-xte(4,k))/dt;
          acc = [accx;accy];
          if accx >= 5
             break
             acc = zeros(2,1);
          elseif accy >= 5
             break
             acc = zeros(2,1);
          end
       end
       tpred(:,l+1) = phi*tpred(:,l)+del*acc;
      end
      nbad = 0;
   end
   %  PLOT OBSERVED AND ESTIMATED TRACKS (TARGET AND OBSERVER)
   plot(xfol(1,1:k+1),xfol(3,1:k+1),'xw',xfo2(1,1:k+1),
   xfo2(3,1:k+1),'xw',...
   xto(1,1:k+1),xto(3,1:k+1),'.b',xte(1,1:k+1),xte(3,1:k+1),'+w',...
   zpos(1,1:k+1),zpos(2,1:k+1),'ow',tpred(1,:),tpred(3,:),'*g',...
   eex,eey,'-r'),grid
   title('ACTUAL/OBSERVED/ESTIMATED/PREDICTED TGT TRKS
   USING BRG/RNG OBS'),...
   xlabel('DISTANCE X'),ylabel('DISTANCE Y')
end
diary off
meta brgrng



% JOHN A. HUCKS II
% FILE NAME : DOPPLER.M
% EXTENDED KALMAN FILTER PROGRAM USING A DOPPLER SHIFTED FREQUENCY
% SIGNAL AND THE BEARING TO THAT SIGNAL.

% THIS PROGRAM SIMULATES THE TRACK GENERATED BY AN EXTENDED KALMAN
% FILTER, WITH INPUTS FROM A FUSED SENSOR ARRAY AS IT TRACKS A TARGET
% ACROSS AN X-Y GRID SPACE (PLRS FLAT EARTH PROJECTION).

clg
clear eex eey xto
!erase doppler.met
diary doppler

%  STATE EQUATIONS

%  OBSERVATION TIME INTERVAL (input from marine.m)

%  STATE TRANSITION MATRIX
phi = [ 1   dt  0   0   0
        0   1   0   0   0
        0   0   1   dt  0
        0   0   0   1   0
        0   0   0   0   1 ];

%  SYSTEM NOISE COEFFICIENT MATRIX
del = [ dt   dt*dt/2   0     0       0
        0       dt     0     0       0
        0       0      dt   dt*dt/2  0
        0       0      0      dt     0
        0       0      0      0      dt ];

%  STATE VARIABLES, FILTER ESTIMATES, TIME, ANI OBSERVED DATA

%  NUMBER OF OBSERVATIONS (input from marine.m)
% ESTIMATED TARGET STATE
xte  = zeros(5,kmax);                        % initial est. target state

% OBSERVER (SENSOR) INITIAL POSITIONS AND VELOCITIES
% (input from marine.m)
xfoa = zeros(4,kmax);
xfoa(:,1) = xfol(:,1);
xfol = xfoa;

% OBSERVER SPEED (input from marine.m)

% INITIAL TIME SETTING
Time = zeros(1,kmax);

% ACTUAL TRACK POSITIONS (input from marine.m)
x = zeros(1,kmax);
y = zeros(1,kmax);
T = 0;
for k = 1
        x(k) = tgt(1,1);
        y(k) = tgt(4,1);
        fo(k) =tgt(7,1);
end
for k = 2:kmax
        T = T+dt;
        x(k) = (0.5*tgt(3,1)*(T^2))+(tgt(2,1).*T)+(tgt(1,1));
        vx(k) = (tgt(3,1)*T)+tgt(2,1);
        y(k) = (0.5*tgt(6,1)*(T^2))+(tgt(5,1).*T)+(tgt(4,1));
        vy(k) = (tgt(6,1)*T)+tgt(5,1);
        fo(k) = tgt(7,1);
end
        zpos = [ x
                 vx
                 y
                 vy
                 fo ];                         % rest frequency xmtr (Hz)

% INPUT OBSERVATION VALUES (BEARING/FREQUENCY)
% INITIAL OBSERVED TARGET STATE
   xto = zeros(5,kmax);
   xto(:,1) = [ tgt(1,1)                       % (km in x direction)
                tgt(2,1)                       % (km/hr in x direction)
                tgt(4,1)                       % (km in y direction)
                tgt(5,1)                       % (km/hr in y direction)
                tgt(7,1) ];                    % rest frequency xmtr (Hz)
   deltlx = zpos(1,:) - xfol(1,:);
   deltly = zpos(3,:) - xfol(3,:);
   brgl =  atan2(deltlx,deltly);               % bearing thtal
   vp = 1.0789e9;                              % spd light (km/hr)
   for k = 1:kmax
       f(k) = (zpos(5,k)*vp)/(vp+((zpos(1,k)*zpos(2,k))+(zpos(3,k)*...
              zpos(4,k)))/sqrt((zpos(1,k)^2+zpos(3,k)^2)));
   end
   zobs = [brgl;f];

   xtol = zeros(5,kmax);
   xtol(1,:) = zpos(1,:);
   xtol(2,:) = zpos(2,:);
   xtol(3,:) = zpos(3,:);
   xtol(4,:) = zpos(4,:);
   xtol(5,:) = zpos(5,:);
   xtol(:,1) = xto(:,1);
   xto = xtol;

% STATE EXCITATION AND MEASUREMENT ERROR COVARIANCE MATRICES
varv  = 1.005;                    % variance of linear acceleration
varth = 5.05;                     % variance of angular acceleration
vfreq = 1.0;                      % variance of the rest frequency

R  = [ 1e-11  0
        0   5e-11 ];                        % measurement noise

% INPUT OBSERVATION VALUES WITH NOISE ADDED
rand('normal')
zobs = zobs + sqrt(R(1,1))*rand(zobs);

% INITIAL KALMAN MATRICES AND OBSERVATION
I = eye(5);
% INITIAL KALMAN P MATRIX
P0 = [ 300   0  0    0  0           % reset value of Error Covariance
        0   300  0   0  0
        0    0  300  0  0
        0    0  0   300  0
        0    0  0    0  300  ];

% REINITIALIZING P MATRIX
P = P0;

% KALMAN ESTIMATE
xte(:,1) = [ xto(1,1)                % (km - x direction)
             tgt(2,1)                % (km/hr)
             xto(3,1)                % (km - y direction)
             tgt(5,1)                % (km/hr)
             tgt(7,1) ];             % rest frequency xmtr (Hz)_

tpred = xte(:,1);

% SIMULATION OF KALMAN FILTER OPERATION
% NUMBER OF BAD OBSERVATIONS
nbad = 0;                            % initial setting bad obs. cntr.
% SIMULATION COUNTER LOOP
k = 0;
for k1=1:10*kmax
   k = k + 1;                        % actual loop counter
   if (k == kmax)                    % check for termination
      break
   end

% UPDATE FORWARD OBSERVER'S/ FAC'S POSITION (IF THEY ARE MOVING)
   if  k <= 3
      way = pi/2;
   else
   way = -(3*pi/4);
end
if spd == 0
   xfol(1,k+1) = xfol(1,k);
   xfol(3,k+1) = xfol(3,k);
else
   xfol(1,k+1) = xfol(1,k) + spd*sin(way)*dt;
   xfol(2,k+1) = spd*sin(way);
   xfol(3,k+1) = xfol(3,k) + spd*cos(way)*dt;
   xfol(4,k+1) = spd*cos(way);
end

% PROJECT AHEAD KALMAN STATE ESTIMATE
xte(:,k+1) = phi*xte(:,k);

% UPDATE Q
vt = sqrt((xte(2,k+1))^2 + (xte(4,k+1))^2);
sigxdd2 = ((xte(2,k+1)/vt)^2*varv) + ((xte(4,k+1)^2)*varth);
sigydd2 = ((xte(4,k+1)/vt)^2*varv) + ((xte(2,k+1)^2)*varth);
sigxydd = (xte(2,k+1)*xte(4,k+1))*((varv/(vt*vt))-varth);
q11 = ((dt*dt)/2)^2*sigxdd2;
q12 = ((dt*dt*dt)/2)*sigxdd2;
q13 = ((dt*dt)/2)*sigxydd;
q14 = ((dt*dt*dt)/2)*sigxydd;
q15 = 0;
q21 = q12;
q22 = (dt*dt)*sigxdd2;
q23 = ((dt*dt*dt)/2)*sigxydd;
q24 = (dt*dt)*sigxydd;
q25 = 0;
q31 = q13;
q32 = q23;
q33 = ((dt*dt)/2)^2*sigydd2;
q34 = ((dt*dt*dt)/2)*sigydd2;
q35 = 0;
q41 = q14;
q42 = q24;
q43 = q34;
q44 = (dt*dt)*sigydd2;
q45 = 0;
q51 = q15;
q52 = q25;
q53 = q35;
q54 = q45;
q55 = (dt*dt)*vfreq;

Q1 = [ q11 q12 q13 q14 q15
       q21 q22 q23 q24 q25
       q31 q32 q33 q34 q35
       q41 q42 q43 q44 q45
       q51 q52 q53 q54 q55 ];

Q = del*Q1*(del');

% PROJECT ERROR COVARIANCE
P = phi*P*(phi') + Q;
```

```
% UPDATE H
r12 = sqrt((xte(1,k+1)^2)+(xte(3,k+1)^2));
r32 = (sqrt((xte(1,k+1)^2)+(xte(3,k+1)^2)))^3;
deltxvy = xte(1,k+1)*xte(4,k+1) - xte(3,k+1)*xte(2,k+1);
deltyvx = xte(3,k+1)*xte(2,k+1) - xte(1,k+1)*xte(4,k+1);
fkvp = xte(5,k+1)*vp;
h11 = -(xte(3,k+1)/((xte(1,k+1)^ 2)+(xte(3,k+1)^2)));
h12 = 0;
h13 = xte(1,k+1)/((xte(1,k+1)^2)+(xte(3,k+1)^2));
h14 = 0;
h15 = 0;
h21 = -((zobs(2,k+1)^2)/fkvp)*(xte(3,k+1)*deltyvx)/r32;
h22 = -((zobs(2,k+1)^2)/fkvp)*(xte(1,k+1)/r12);
h23 = -(zobs(2,k+1)/fkvp)*(xte(1,k+1)*deltxvy)/r32;
h24 = -((zobs(2,k+1)^2)/fkvp)*(xte(3,k+1)/r12);
h25 = zobs(2,k+1)/xte(5,k+1);

H = [   h11 h12 h13 h14 h15
        h21 h22 h23 h24 h25   ];

% GET NEW MEASURED ESTIMATE
deltlx = xte(1,k+1) - xfol(1,k+1);
deltly = xte(3,k+1) - xfol(3,k+1);
zhat1 =  atan2(deltlx,deltly);                % bearing estimate thtal
zhat2 = (xte(5,k+1)*vp)/(vp+((xte(1,k+1)*    % frequency estimate
xte(2,k+1))+(xte(3,k+1)*...
xte(4,k+1)))/sqrt((xte(1,k+1)^2+xte(3,k+1)^2)));
zhat   = [ zhat1
           zhat2 ];

% COMPUTATION OF KALMAN GAIN
vresid = (H*P*(H') + R);                  % variance of the residual
G = P*(H') / vresid;                      % Kalman gain

% COMPUTATION OF UPDATED ERROR COVARIANCE
P = (I - G*H)*P;

% COMPUTATION OF RESIDUAL
resid = zobs(:,k+1) - zhat;

% IF RESIDUAL IS TOO BIG, THEN RESET P AND DO CALCULATIONS AGAIN.
if (abs(resid(1)) > 1.0*sqrt(abs(vresid(1,1))) ...
      ] abs(resid(2)) > 1.0*sqrt(abs(vresid(2,2))) ...
      & nbad == 0)
   nbad = 1;
   k = k-1;
   P = P0;                                % reset P
else

% COMPUTATION OF UPDATED ESTIMATE
   xte(:,k+1) = xte(:,k+1) + G*resid;

% ERROR ELLIPSOID PLOTTING ROUTINE
   M = [xte(1,k+1) xte(3,k+1)];
   K = [ P(1,1) P(1,3)
```

```
            P(3,1) P(3,3) ];
      [xp,yp] = errellip(M,K,.60,25,0);
      eex(:,k) = xp;
      eey(:,k) = yp;

   % PREDICTED (FUTURE TRACK) USING KALMAN ESTIMATE
      tpred(:,1) = xte(:,k+1);
      acc = zeros(5,1);
      for l = 1:kmax
          if k >= 3
             acc(2,1) = (xte(2,k+1)-xte(2,k))/dt;
             acc(4,1) = (xte(4,k+1)-xte(4,k))/dt;
             if acc(2,1) >= 5
                break
                acc = zeros(5,1);
             elseif acc(4,1) >= 5
                break
                acc = zeros(5,1);
            end
          end
          tpred(:,l+1) = phi*tpred(:,l)+del*acc;
      end
      nbad = 0;
      end
   % PLOT OBSERVED AND ESTIMATED TRACKS (TARGET AND OBSERVER)
   plot(xfol(1,1:k+1),xfol(3,1:k+1),'xw',xto(1,1:k+1),
   xto(3,1:k+1),'.b',...
   xte(1,1:k+1),xte(3,1:k+1),'+w',zpos(1,1:k+1),zpos(3,1:k+1),'ow',...
   tpred(1,:),tpred(3,:),'*g',eex,eey,'-r'),grid
   title('ACTUAL/OBSERVED/ESTIMATED/PREDICTED TGT TRKS
   USING DOPPLER '),...
   xlabel('DISTANCE X'),ylabel('DISTANCE Y')
end
meta doppler



function    [x,y] = errellip(M,K,Pr,n,MD)
%
% errellip(M,K,Pr,n,MD)
%
%     This function takes the following arguments:
%
%        M     Mean Vector (2 x 1)
%        K     Covariance Matrix (2 x 2)
%        Pr    Probability (0 .. 1)
%        n     Number of points to compute
%        MD    Compute by Mahalanobis Distance vice probability
%              0 = Probability
%              1 = Mahalanobis Distance
%
%  and returns x and y vectors of the confidence ellipse for the
%  given probability or Mahalanobis Distance.

%  Stephen L. Spehn     15 Nov 1989
```

```
if nargin  = 5
   error('Incorrect number of arguments to errellip.');
end;
if MD == 1
   c = abs(Pr);
else
   Pr = max(min(.995,Pr),.005);

   %  Cubic spline fit for the ellipse constant.
   %  Using this method is a compromise between speed and accuracy.
   pp = [ 12; .005; .01; .025; .05; .1; .25; .5; .75; .9;
         .95; .975; .99; .995; 4.0;
   4.190809197869072e+1;  4.190809197872625e+1;  -2.118721291999464e+1;
   3.970929262330003;     2.15930349191602;      5.955793735647319e-1;
   1.62882241111034e+1;   8.243758565140l9e+1;   2.725551521454689e+3;
   4.176985755223079e+2;  2.087990965023806e+5;  2.087990965023842e+5;
   -3.810356328008524e-1; 2.475857468793011e-1;  2.133449885921905;
   5.444089169224142e-1;  1.140048306271903;     2.111734877634124;
   2.55841940780766;      1.477458749113522e+1;  5.187150103426589e+1;
   4.6070422925247e+2;    4.920316224166436e+2;  9.887990965023757e+3;
   2.02085747586453?;     2.02019022643493;      2.055905760926949;
   2.122852230998054;     2.20707509215777;      2.694842569743673;
   3.862381141104123;     8.195632865839841;     1.819254614465004e+1;
   4.382133265898673e+1;  6.763972895071458e+1;  2.23340067762321e+2;
   .01; .0201; .0506; .103; .211; .575;
   1.39; 2.77; 4.61; 5.99; 7.38; 9.21 ];
   c = ppval(pp,Pr);
end;

%  Get Eigenvectors, Eigenvalues, and translated variances
[Evec,Eval] = eig(K);
sigx = sqrt(Eval(1,1));
sigy = sqrt(Eval(2,2));

%  Parameterized ellipse equations
t = 0:(2*pi/n):(2*pi);
xv = sigx*c*cos(t);
yv = sigy*c*sin(t);

%  Translate back to the original coordinates
x = (xv*Evec(1,1) + yv*Evec(1,2) + M(1))';
y = (xv*Evec(2,1) + yv*Evec(2,2) + M(2))';



function y = reshape(x,m,n)
%RESHAPE RESHAPE(X,M,N) returns the M-by-N matrix whose elements
%are taken columnwise from X.  An error results if X does
%not have M*N elements.
mm,nn = size(x);
if mm*nn  = m*n
error('Matrix must have M*N elements.')
end
```

```
y = zeros(m,n);
y(: ) = x;
```

LIST OF REFERENCES

1. Sorenson, H.W., "Kalman Filtering Techniques", *Advanced Control Systems*, Vol. 3, Chap. 5, 1966.

2. Tzu, Sun. *The Art of War*, Clavell, J., ed., Delacorte Press, NY, 1983.

3. U.S. Army Communications and Electronics Command, Contract No. DAAB07-83-C-J031, *PLRS System Technical Description*, Fort Monmouth, NJ, April 1986.

4. Mitschang, G.W., "An Application of Nonlinear Filtering Theory to Passive Target Location and Tracking", Doctoral Dissertation, Naval Postgraduate School, Monterey, CA, June 1974.

5. Collin, R.E. *Antennas and Radio Wave Propagation*, p. 307, McGraw-Hill, NY, 1985.

6. Gelb, A., and others, *Applied Optimal Estimation*, Gelb, A., ed., p. 182, MIT Press, MA, 1974.

7. Therrien, C.W., *Decision Estimation and Classification*, John Wiley and Sons, Inc., NY, 1989.

8. Baheti, R.S., "Efficient Approximation of Kalman Filter for Target Tracking", *IEEE Trans. on Aerospace and Electronic Systems*, Vol. AES-22, No. 1, pp. 8-14, 1986.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Commandant of the Marine Corps Code TE 06 Headquarters, U. S. Marine Corps Washington, D.C. 20380-0001	1
4.	Chairman, Code 62 Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Professor Harold A. Titus, Code 62Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	2
6.	Professor Roberto Cristi, Code 62Cx Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
7.	Professor Mostafa Ghandahari, Code 53Gh Department of Mathematics Naval Postgraduate School Monterey, CA 93943-5000	1
8.	Commanding Officer Attn: CAPT J.A. Hucks II USMC Marine Corps Tactical Systems Support Activity Marine Corps Base Camp Pendleton, CA 92055-6045	5